云存储应用技术

第六章：分布式云存储

丁烨
dingye@dgut.edu.cn
网络空间安全学院

2019-12-12

東莞理工學院
DONGGUAN UNIVERSITY OF TECHNOLOGY

目录

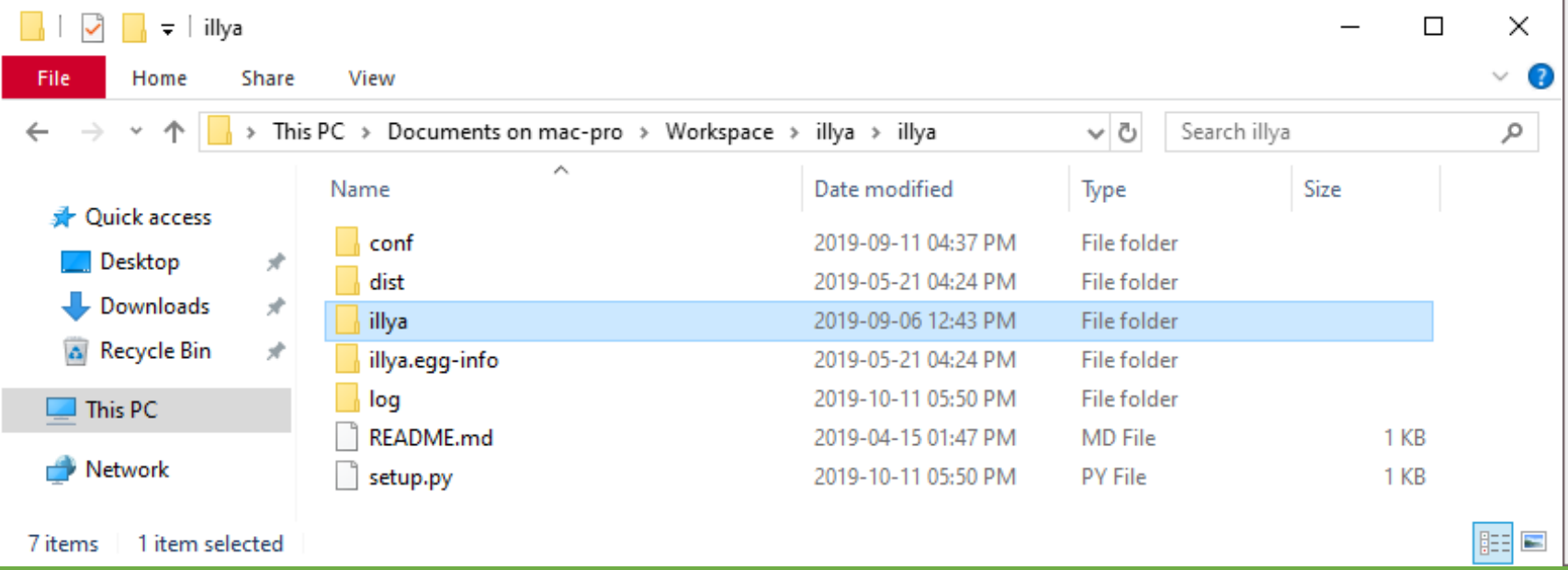
分布式云存储概述

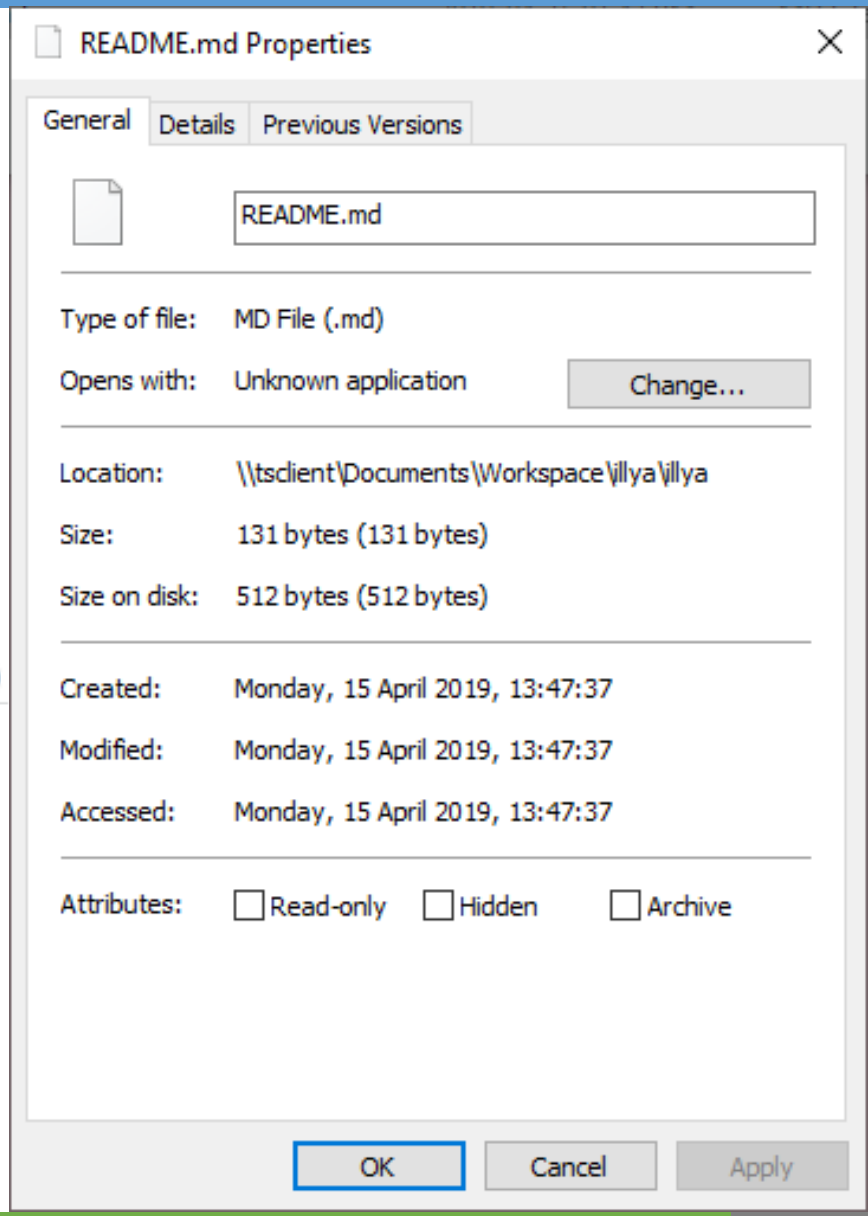
文件系统

- 文件系统（File System）
- 计算机的文件系统是一种存储和组织计算机数据的方法
- 文件系统使用“文件（File）”和树形“目录（Directory）”的抽象逻辑概念代替了物理设备使用数据块（Block）的概念，用户使用文件系统来保存数据不必关心数据实际保存在物理设备的地址为多少的数据块上，只需要记住这个文件的所属目录和文件名
- 严格地说，文件系统是一套实现了数据的存储、分级组织、访问和获取等操作的抽象数据类型（Abstract Data Type）

❖ Windows 资源管理器

❖ 在 Windows 中，Windows 资源管理器可以查看和管理目录（“文件夹（Folder）”）和文件





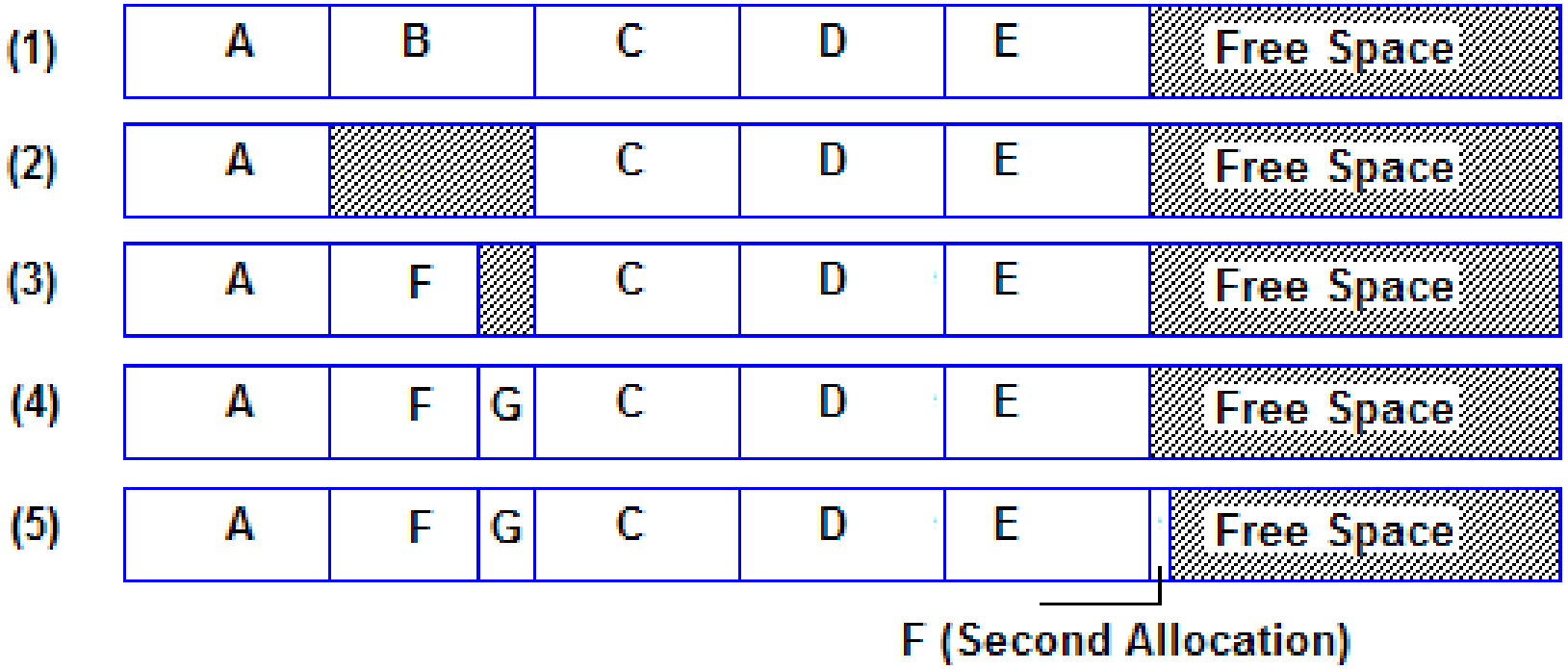
- UNIX Shell
- 在 UNIX 中，可以使用 Shell 查看和管理目录和文件
- 查看当前目录下的文件：ls -al 或 ll
- 以树形结构产看当前目录结构：tree .

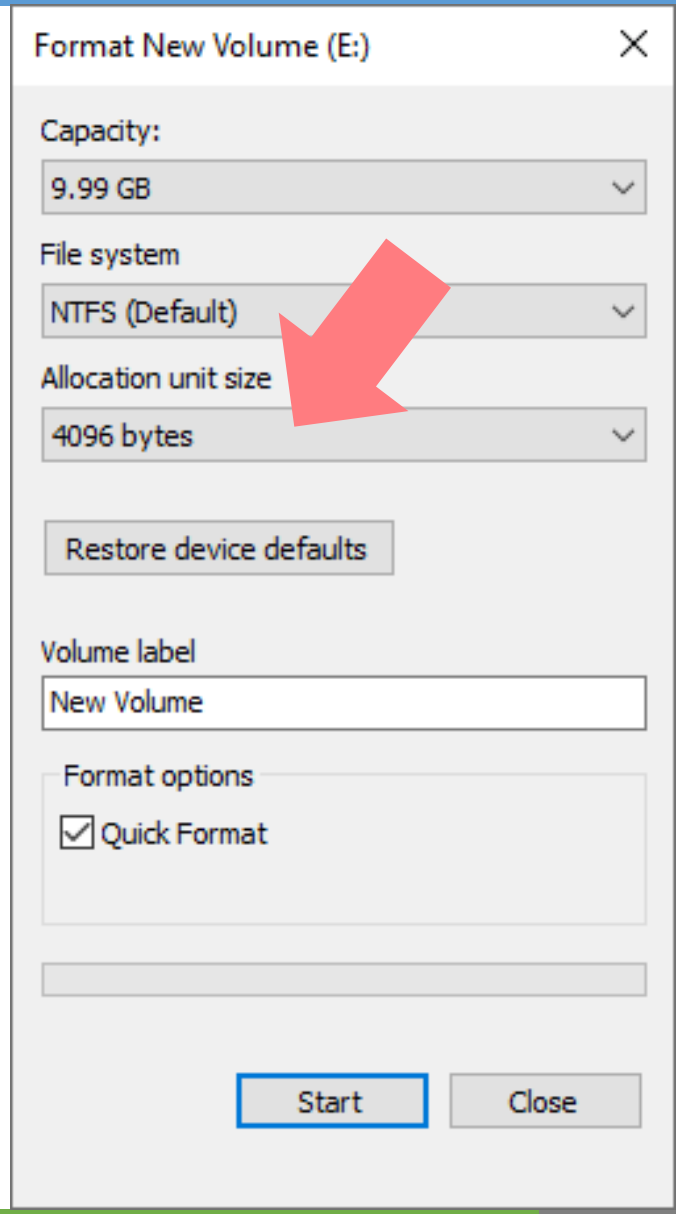
```
~/Documents/Workspace » tree .
.
`-- docker-sshd
    |-- CentOS
    |   `-- Dockerfile
    |-- LICENSE
    |-- README.md
    `-- Ubuntu
        `-- Dockerfile

3 directories, 4 files
```

```
~/Documents/Workspace/docker-sshd(master) » ll
total 16K
drwxrwxr-x 2 valency valency 4.0K Oct  3 19:01 CentOS
-rw-rw-r-- 1 valency valency 1.1K Oct  3 19:01 LICENSE
-rw-rw-r-- 1 valency valency  602 Oct  3 19:01 README.md
drwxrwxr-x 2 valency valency 4.0K Oct  3 19:01 Ubuntu
```

- 文件的组织形式
- 在文件系统中，文件会被分割成数据块（Block）存放在物理设备中
- 一个文件可能存放于多个不同的物理位置中





分布式云存储概述

文件系统

- 文件的组织形式
- 文件的分割信息会被作为元数据（Metadata）存放于分区（Partition）中
- 目录信息作为额外的元数据也会被存放于分区中

- 由于元数据的存在，不同文件可以指向同一部分数据块
- 这种存储方式叫做“文件链接（link）”（通常在 UNIX 中） 或“快捷方式（Shortcut）”（通常在 Windows 中）

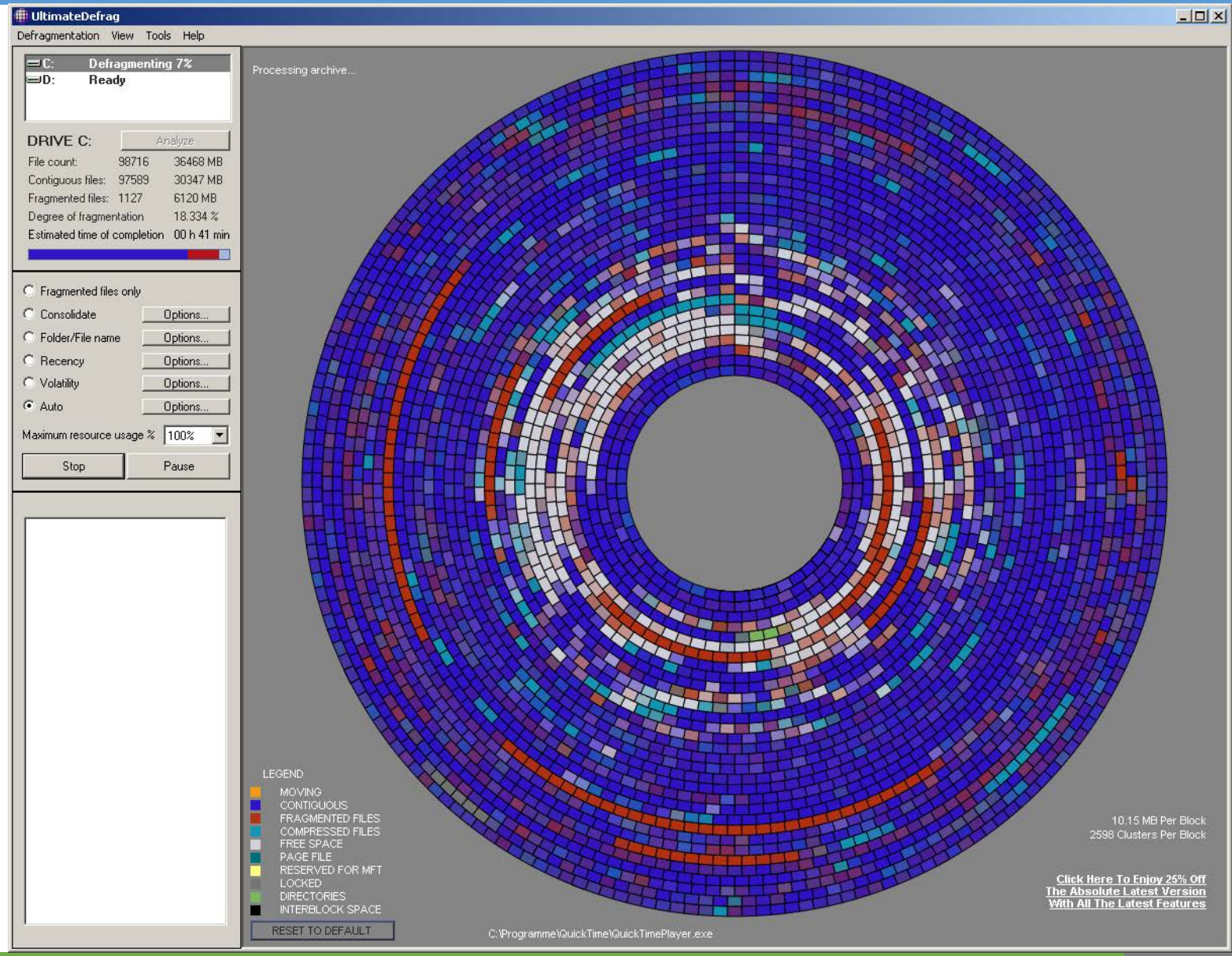
- 分区有很多种组织形式和格式，例如 FAT32、NTFS，APFS 等
- 创建某一种分区格式被称为“格式化（Format）”
- 一个物理设备可以创建多个分区，分区由分区表（Partition Table）管理并保存

- 一般文件系统的访问方式
- 从物理设备中读取文件时，一般会按照以下顺序进行：

- 首先，从物理设备中读取分区表，确定分区表的结构
- 其次，从分区表中定位到某个分区，读取分区的格式和元数据
- 从分区中读取目录信息（如需要）
- 从分区中读取文件信息，并定位文件在物理设备中的数据块
- 从不同的数据块中读取数据
- 合并数据（如需要）并存放于内存中供应用程序使用

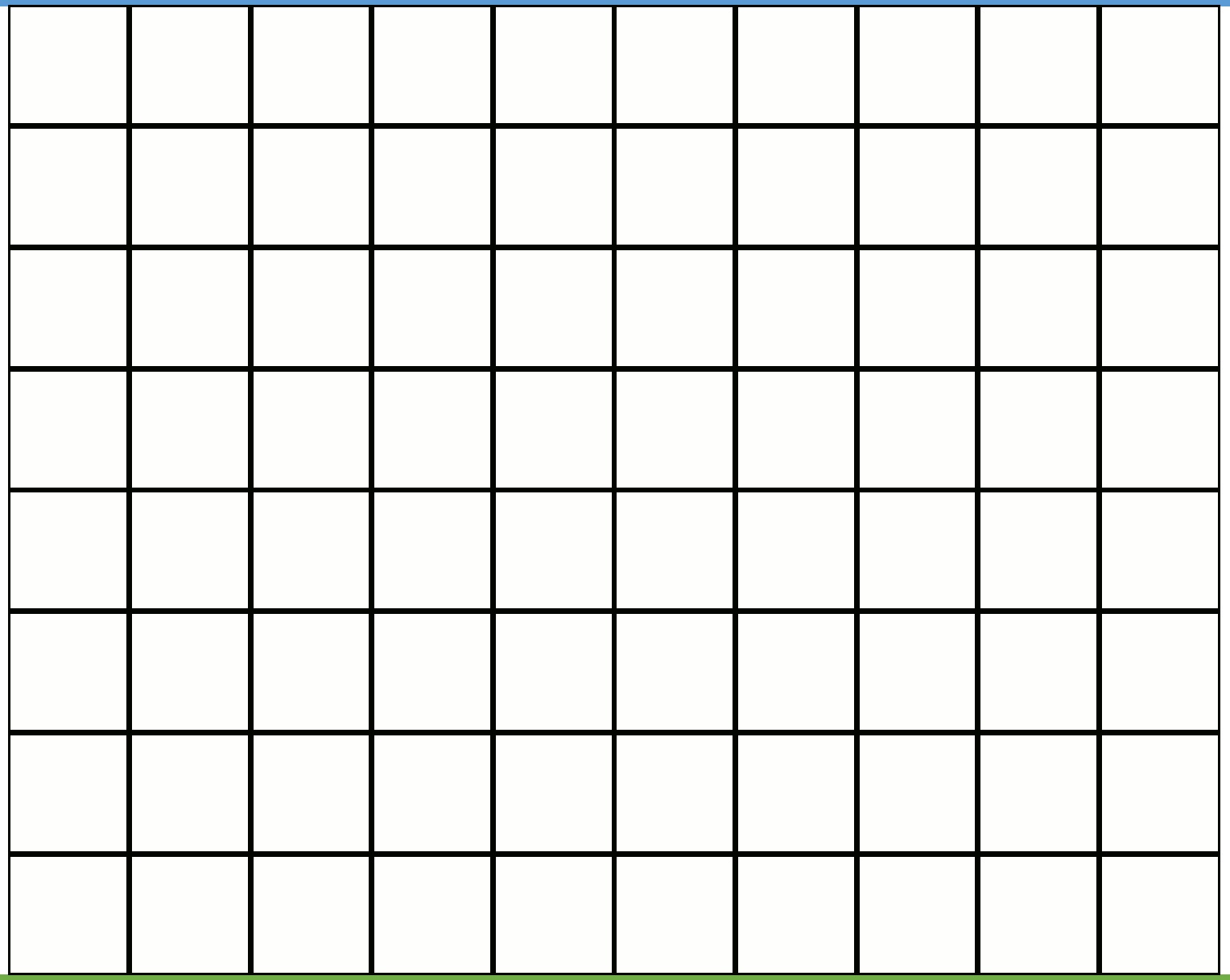
文件系统的细节

- 文件系统碎片（File System Fragmentation）
- 文件系统将文件内容非连续排列以允许就地修改其内容的后果之一
- 在传统的机械硬盘中，磁盘碎片会增加磁盘磁头移动的频率，即增加了寻道时间，这会降低磁盘读写性能，进而影响操作系统及软件的性能
- 对现有碎片的更正称为碎片整理，是将文件和可用空间重新组织为连续区域的过程



分布式云存储概述

文件系统的细节

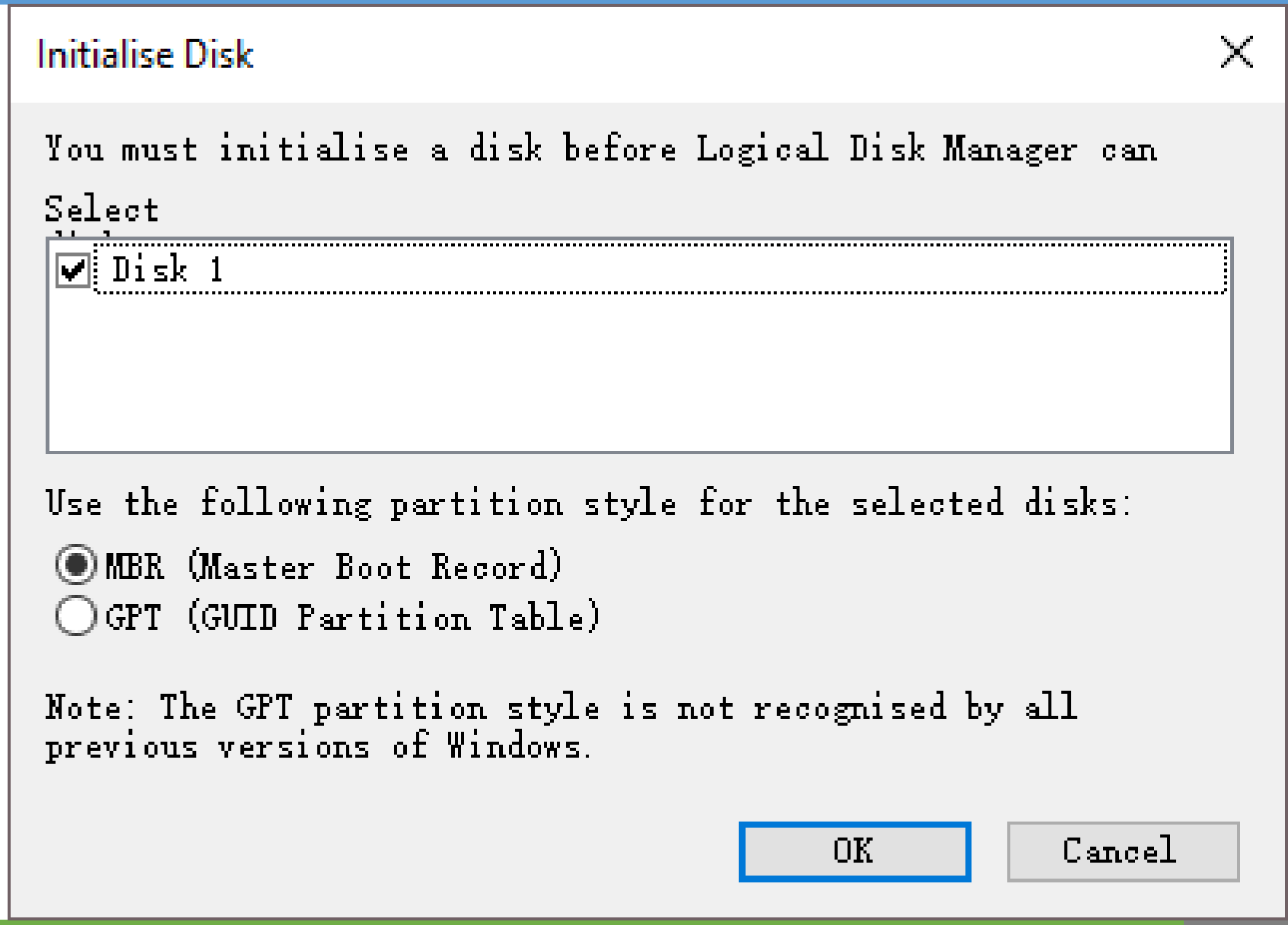


文件系统的细节

- 文件系统碎片（File System Fragmentation）
- 对于机械硬盘而言，磁盘碎片是个很严重的问题，因此，碎片是在文件系统的研究与设计的一个重要问题。碎片的遏制不仅很大程度上依赖于文件系统在磁盘上的格式，还取决于它的实现
- 对于固态硬盘（SSD）而言，由于没有“磁盘寻道”，且固态硬盘也不是“旋转的”，所以没有磁盘碎片的问题
- 对固态硬盘进行碎片整理反而会缩短其使用寿命

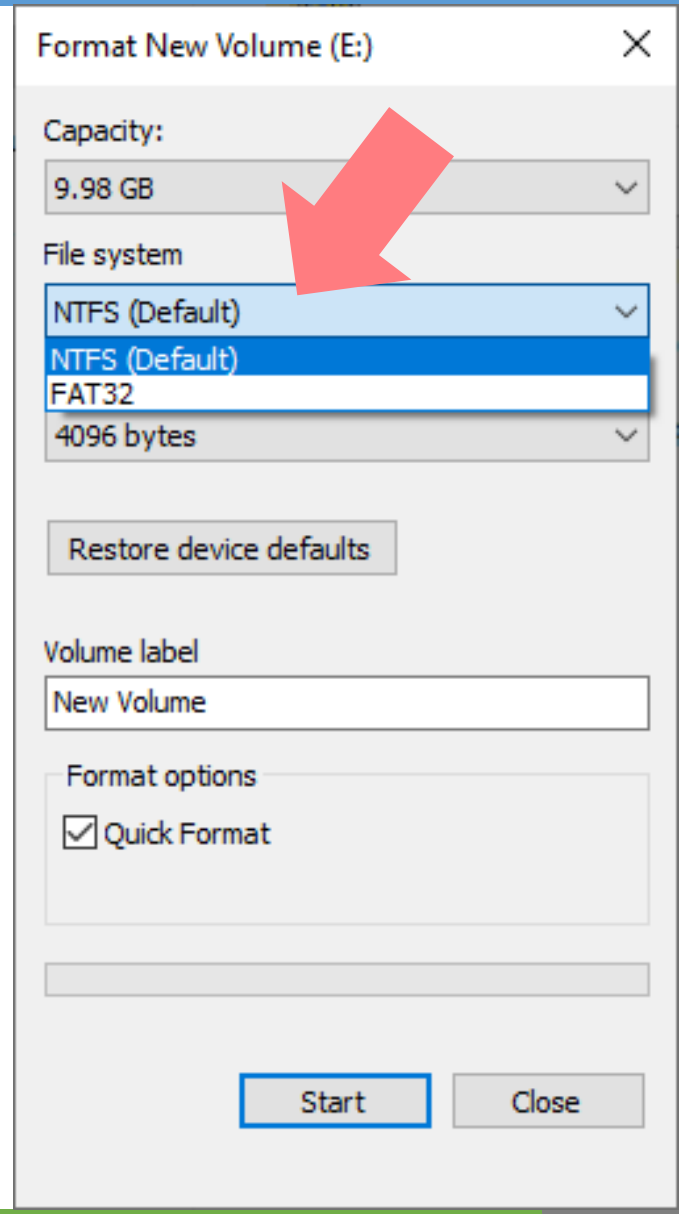
文件系统的细节

- 分区表（Partition Table）
- 分区的组织形式，常见的有：
- 主引导记录
- Master Boot Record，MBR
- 全局唯一标识分区表
- GUID Partition Table，GPT
- Windows 10 / macOS Catalina 一般采用 GPT



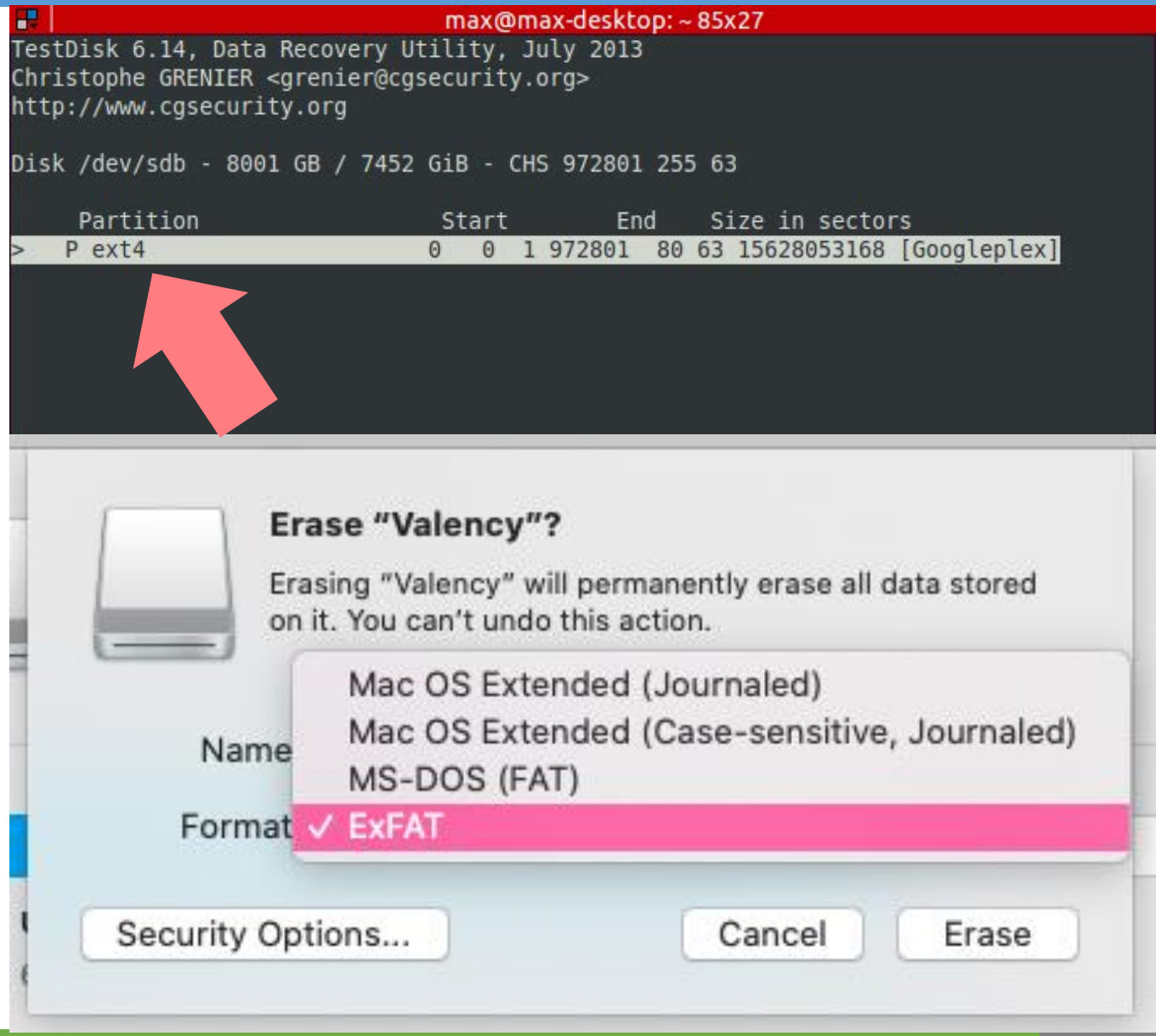
文件系统的细节

- 分区格式
- 分区中文件和目录的组织形式，Windows 中常见的有：
- FAT / FAT32（File Allocation Table）
- NTFS（New Technology File System）
- exFAT（Extensible File Allocation Table）



文件系统的细节

- 分区格式
- UNIX 中常见的有：
- ext2 / ext3 / ext4（Extended File System）
- APFS（Apple File System）
- exFAT（Extensible File Allocation Table）



文件系统的细节

- 可移植操作系统接口（Portable Operating System Interface，POSIX）
- 是 IEEE 为要在各种 UNIX 操作系统上运行软件，而定义 API 的一系列互相关联的标准的总称，其正式称呼为 IEEE Std 1003，而国际标准名称为 ISO/IEC 9945
- POSIX 这个名称是由理查德斯托曼（Richard Matthew Stallman，RMS，GNU 之父）应 IEEE 的要求而提议的一个易于记忆的名称，其中 X 表明其对 UNIX API 的传承
- Linux 基本上逐步实现了 POSIX 兼容，但并没有参加正式的 POSIX 认证
- 微软的 Windows NT 声称部分实现了 POSIX 标准
- 当前的 POSIX 主要分为四个部分：Base Definitions、System Interfaces、Shell and Utilities、Rationale
- 符合 POSIX 标准的文件系统通常才能为大部分应用程序所兼容

分布式云存储概述

分布式文件系统

- 分布式文件系统（Distributed File System）
- 是指文件系统管理的物理存储资源不一定直接连接在本地节点（Node，可简单的理解为一台计算机）上，而是通过计算机网络与多个节点相连
- 类似 RAID 和 LVM，分布式文件系统内部可能包含了多个物理存储资源
- 对于应用程序来言，分布式文件系统应当在许多方面实现“透明性”，也就是说，它们的目标是让应用程序“看不见”它们，这些应用程序看到的是一个类似于本地文件系统（即符合 POSIX 标准）的文件系统
- 在幕后，分布式文件系统处理定位文件、传输数据，并可能提供一些其他功能

分布式云存储概述

分布式文件系统

- 分布式文件系统的设计目标
- 访问透明性：客户端感受不到文件是分布式的，并且可以像访问本地文件一样访问它们
- 位置透明性：存在一致的名字空间，包含本地文件和远程文件，文件名不提供其位置
- 并发透明性：所有客户端都具有相同的文件系统状态视图。这意味着，如果一个进程正在修改一个文件，那么同一系统或远程系统上访问该文件的任何其他进程都会以一致的方式看到修改
- 失败透明性：客户端和客户端程序应在服务器发生故障后正常运行
- 异构性：文件服务应该跨不同的硬件和操作系统平台提供
- 可扩展性：文件系统应该在小型环境中运行良好，并且可以优雅的扩展到更大的环境中（数百到数万个服务器）
- 复制透明性：客户端不应感受到文件复制是跨多个服务器执行的
- 迁移透明性：文件在不同的服务器之间的移动不应该让客户端感受到

分布式云存储概述

分布式文件系统

- 分布式文件系统的优点
- 可以存放极大量的数据
- 可以避免本地磁盘错误
- 物理设备可以不存放于同一个物理地点，容灾性更强
- 易于不同的应用程序共享存储内容
- 相对于文件托管服务而言，透明性更好（即支持 POSIX，且可挂载到本地）

分布式文件系统

- 常见的分布式文件系统
- HDFS（Hadoop Distributed File System）
- Ceph
- Alluxio
- IPFS（InterPlanetary File System）
- Lustre
- Gluster

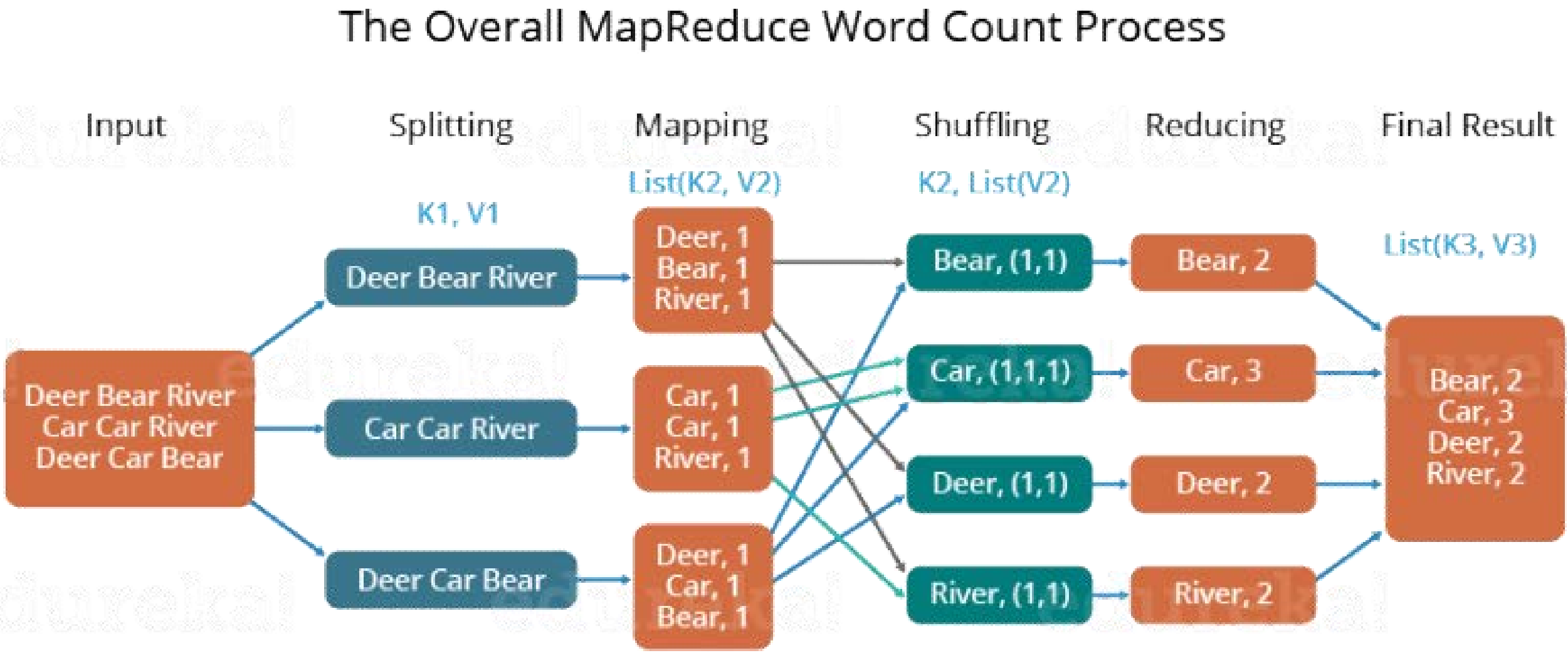
目录

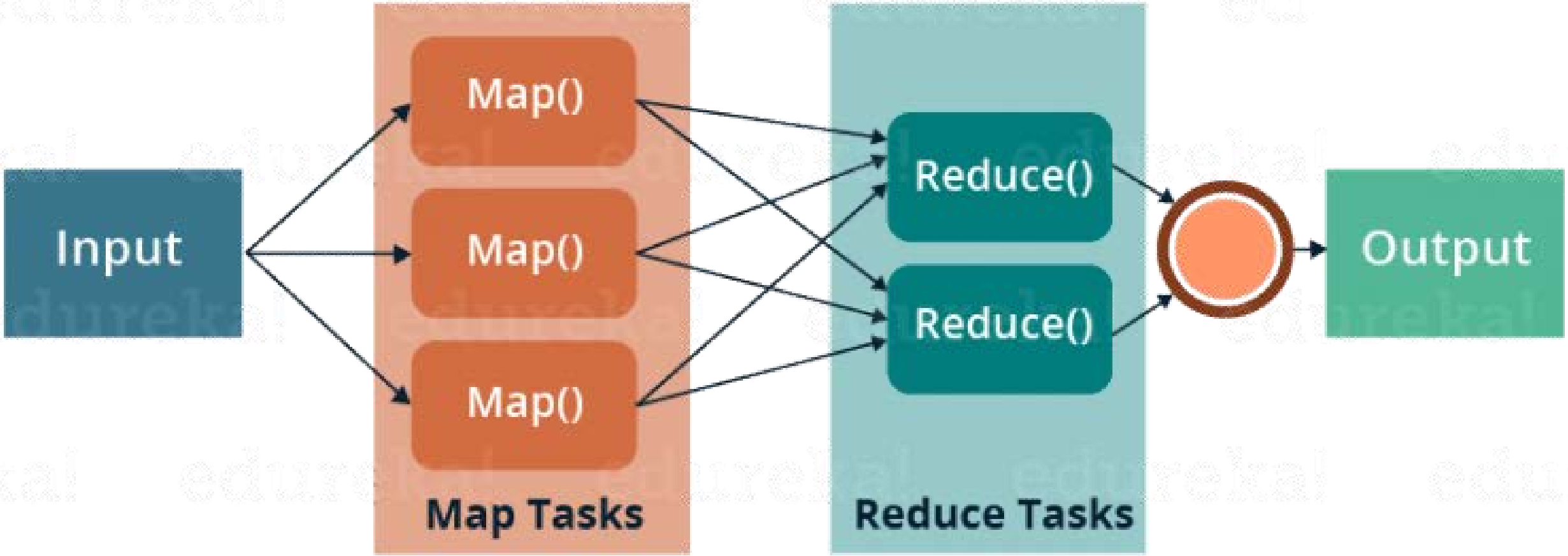
Hadoop 简介

- Apache Hadoop
- http://hadoop.apache.org/

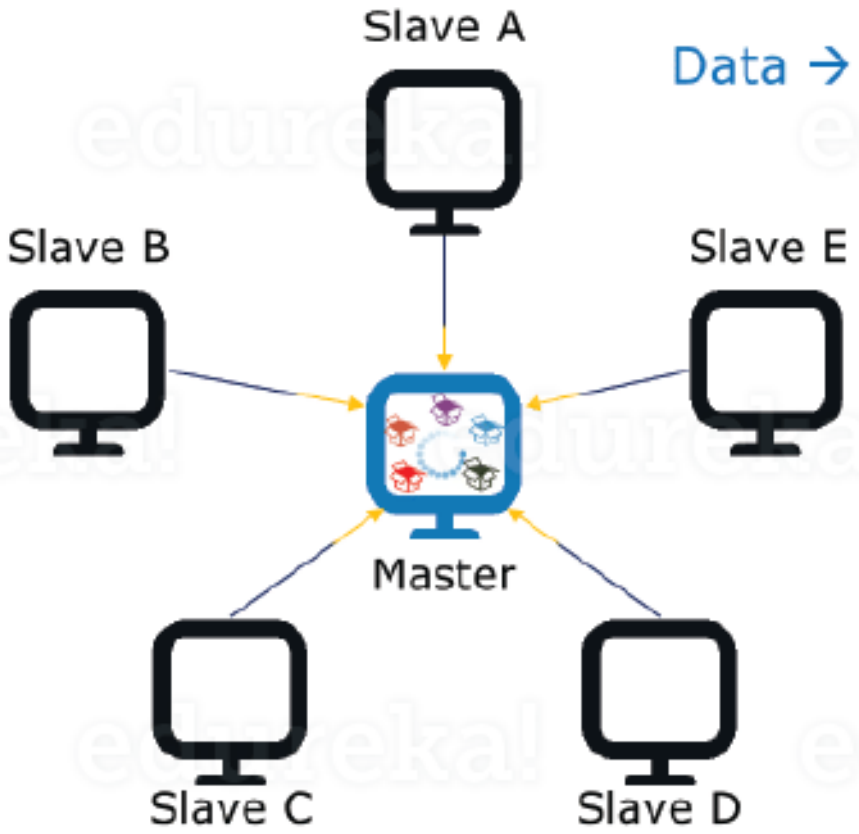
- 一款支持数据密集型分布式应用程序并以 Apache 2.0 许可协议发布的开源软件框架
- Hadoop 是根据 Google 发表的 MapReduce 和 Google 文件系统（GFS）的论文自行实现而成：https://research.google.com/archive/gfs-sosp2003.pdf
- 所有的 Hadoop 模块都有一个基本假设，即硬件故障是常见情况，应该由框架自动处理
- Hadoop 框架透明的为应用提供可靠性和数据移动
- 它实现了名为 MapReduce 的编程范式：应用程序被分割成许多小部分，而每个部分都能在集群中的任意节点上运行或重新运行

- MapReduce
- Google 提出的一个软件架构，用于大规模数据集（大于 1 TB）的并行运算
- 概念“Map（映射）”和“Reduce（归纳）”，及他们的主要思想，都是从函数式编程语言借来的，还有从矢量编程语言借来的特性
- 当前的软件实现是指定一个 Map（映射）函数，用来把一组键值对映射成一组新的键值对，指定并发的 Reduce（归纳）函数，用来保证所有映射的键值对中的每一个共享相同的键组

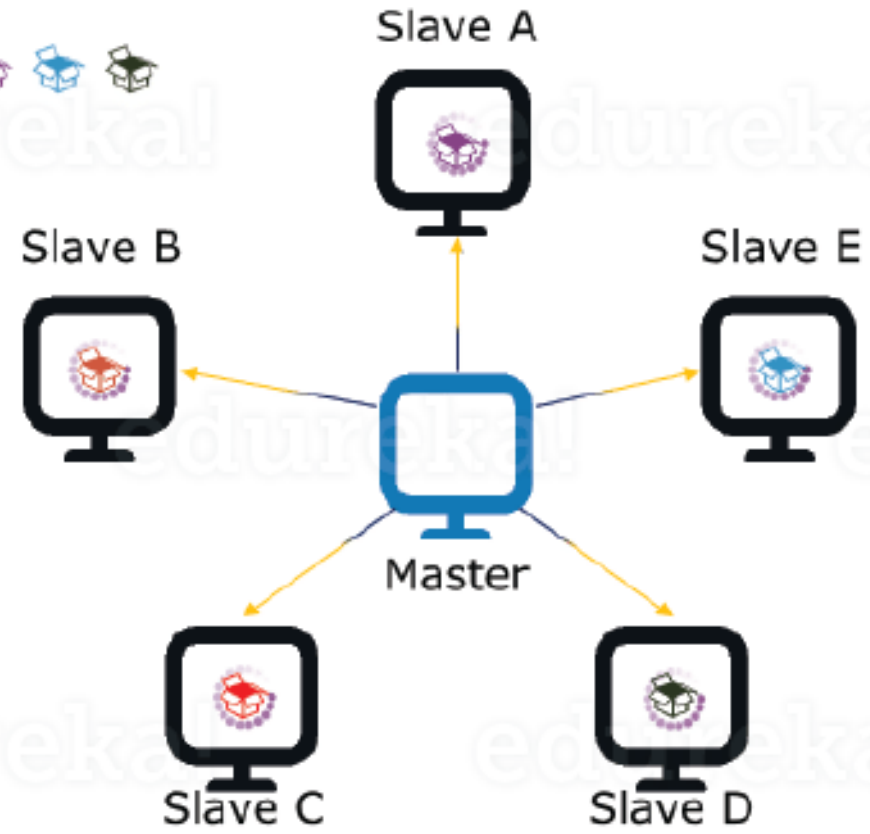

The Overall MapReduce Word Count Process
Input
Splitting
Mapping
Shuffling
Reducing
Final Result
K1, V1
List(K2, V2)
K2, List(V2)
List(K3, V3)
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, (1,1)
Car, (1,1,1)
Deer, (1,1)
River, (1,1)
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2


Input
Map()
Map()
Map()
Map Tasks
Reduce()
Reduce()
Reduce Tasks
Output

Hadoop 简介



1. Moving data to the Processing Unit
(Traditional Approach)



2. Moving Processing Unit to the data
(MapReduce Approach)

Hadoop 简介

- Hadoop 分布式文件系统（Hadoop Distributed File System，HDFS）
- Hadoop 提供了分布式文件系统，用以存储所有计算节点的数据
- 这为整个集群带来了非常高的带宽
- MapReduce 和分布式文件系统的设计，使得整个框架能够自动处理节点故障
- 它使应用程序与成千上万的独立计算的计算机和 PB 级的数据连接起来
- 现在普遍认为整个 Apache Hadoop“平台”包括 Hadoop 内核、MapReduce、Hadoop 分布式文件系统（HDFS）以及一些相关项目，有 Apache Hive 和 Apache HBase 等

HDFS（Hadoop 分布式文件系统）

安装及使用 HDFS

- Hadoop 的安装较为复杂，可以参考 Hadoop 官方教程：
- https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html

Apache Hadoop

The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

Learn more » Download » Getting started »

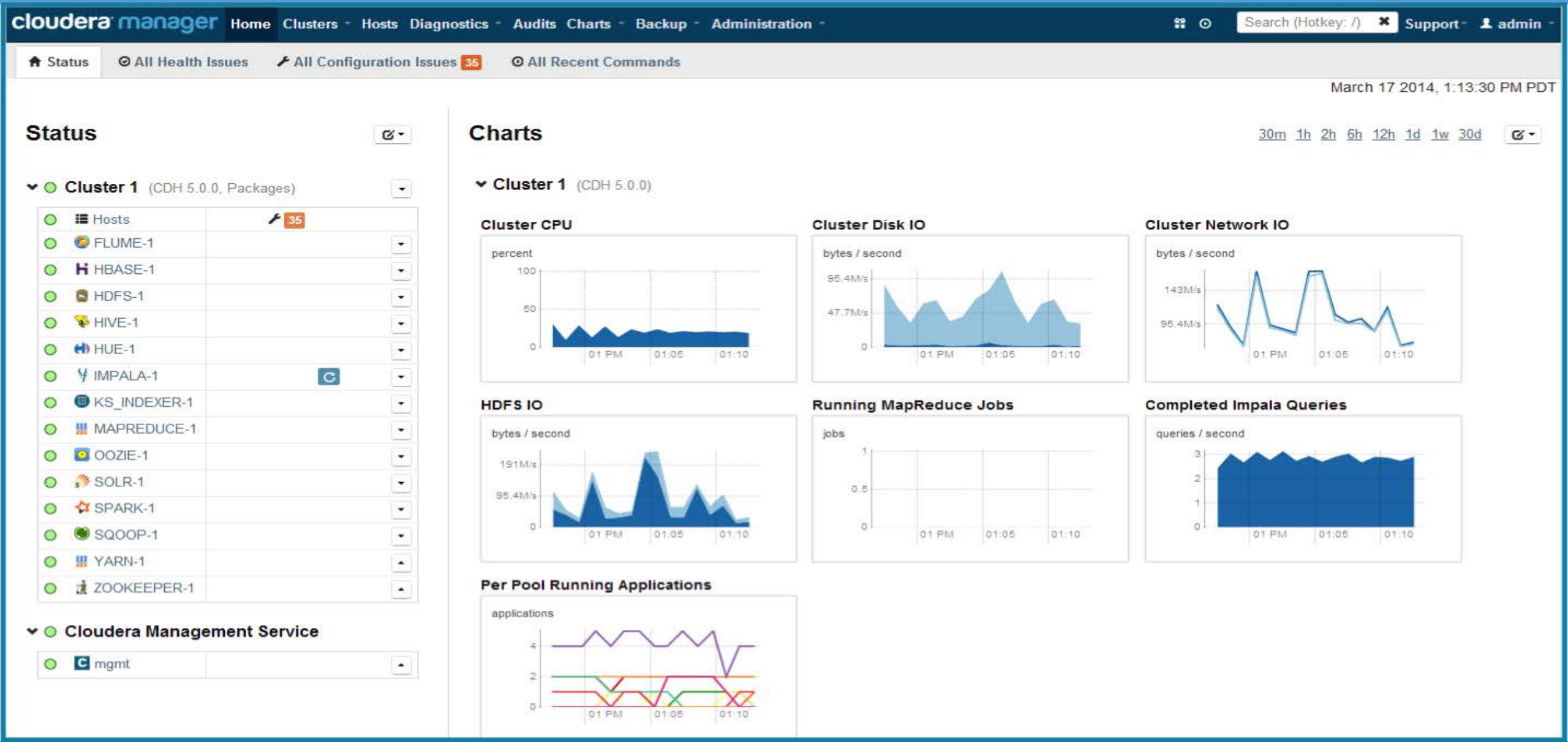
安装及使用 HDFS

- Cloudera
- https://www.cloudera.com/

- 由于 Hadoop 安装过于复杂，工业界普遍使用更为简单的 Cloudera 数据平台（Data Platform）
- Cloudera 提供了一整套自动化安装和配置系统，安装过程全程可视化
- 另外还提供了一套监控（Provision & Monitor）系统，可以远程监控和管理 Hadoop 及其相关的应用平台

HDFS（Hadoop 分布式文件系统）

安装及使用 HDFS

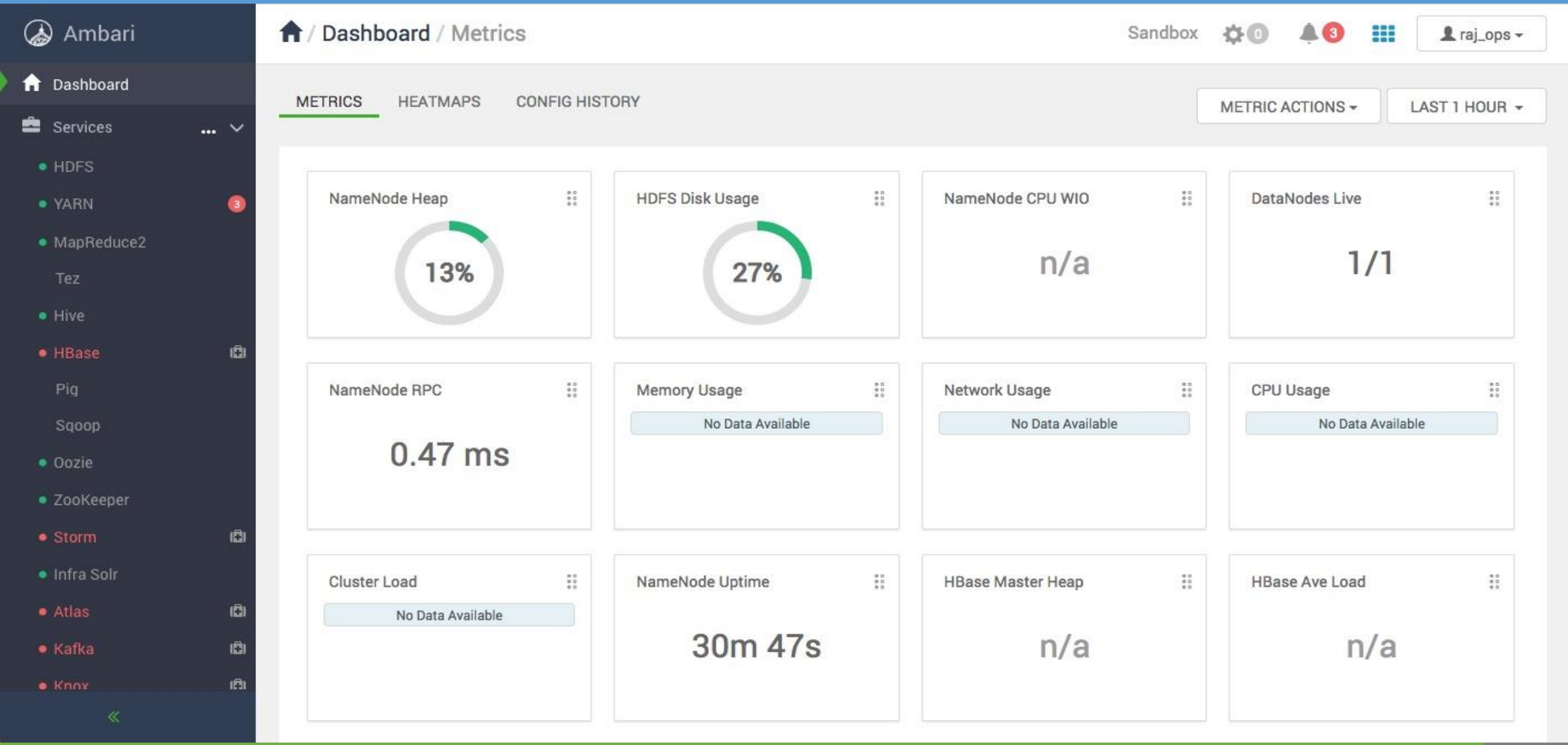


安装及使用 HDFS

- Ambari
- https://ambari.apache.org/

- 开源免费的 Cloudera 替代品
- 功能少于 Cloudera
- 提供基础的 Hadoop 及其相关应用平台的可视化安装、管理、监控等功能

安装及使用 HDFS


Ambari
Dashboard
Services
HDFS
YARN
MapReduce2
Tez
Hive
HBase
Pig
Sqoop
Oozie
ZooKeeper
Storm
Infra Solr
Atlas
Kafka
Knox
Dashboard / Metrics
Sandbox
raj_ops
METRICS
HEATMAPS
CONFIG HISTORY
METRIC ACTIONS
LAST 1 HOUR
NameNode Heap
13%
HDFS Disk Usage
27%
NameNode CPU WIO
n/a
DataNodes Live
1/1
NameNode RPC
0.47 ms
Memory Usage
No Data Available
Network Usage
No Data Available
CPU Usage
No Data Available
Cluster Load
No Data Available
NameNode Uptime
30m 47s
HBase Master Heap
n/a
HBase Ave Load
n/a

安装及使用 HDFS

- Hadoop Docker Container
- https://github.com/big-data-europe/docker-hadoop

- Hadoop / HDFS 的 Docker Container 很多，Big Data Europe 只是其中一个
- 由于 Hadoop 为分布式系统，使用 Hadoop 需要同时启动至少两个 Docker Container
- Big Data Europe 采用了 Docker Compose 进行更方便的部署

安装及使用 HDFS

- Docker Compose
- https://docs.docker.com/compose/

- Docker Compose 是一个用来管理 Docker Container 的工具
- 使用 Docker Compose 可以同时部署、启动、关闭多个 Docker Container
- Docker Compose 相当于管理 Docker Container 的“脚本工具”

A `docker-compose.yml` looks like this:

```
version: '3'
services:
  web:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - .:/code
    - logvolume01:/var/log
    links:
    - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

安装及使用 HDFS

- 使用 Big Data Europe 的 Hadoop Docker Container
- https://clubhouse.io/developer-how-to/how-to-set-up-a-hadoop-cluster-in-docker/

- 首先使用 Git 复制 Big Data Europe 的 Hadoop Docker Container 的 Docker Compose 配置文件：
- `git clone git@github.com:big-data-europe/docker-hadoop.git`

安装及使用 HDFS

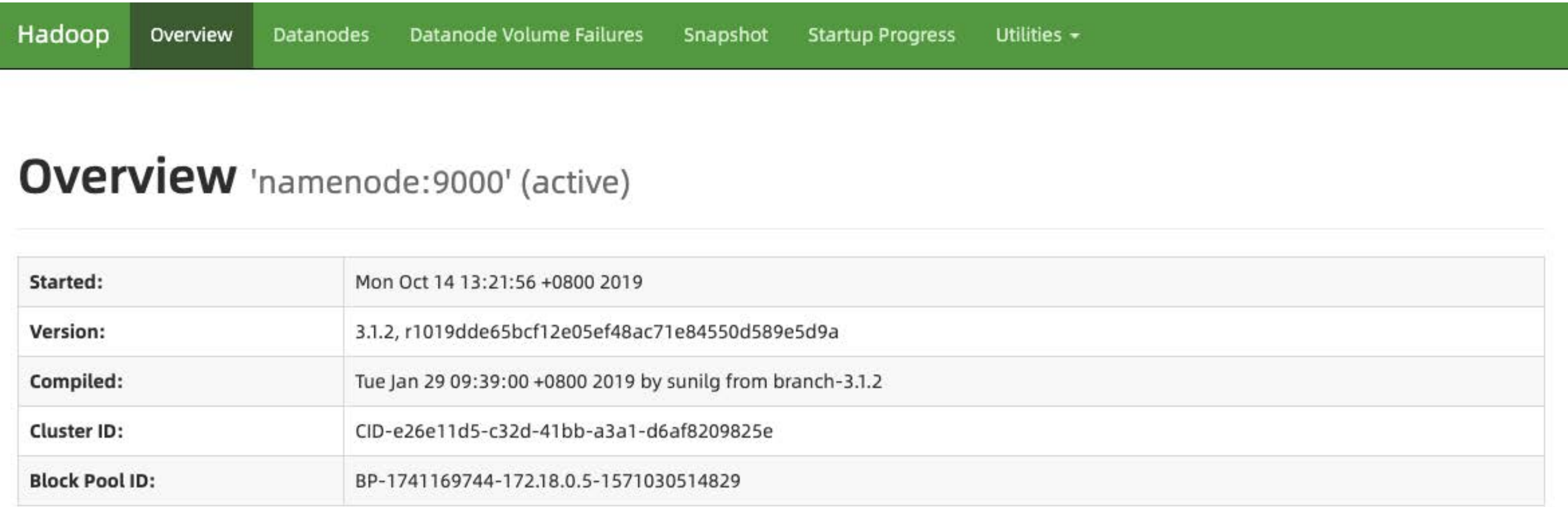
- 通过 Docker Compose 启动本地 Hadoop 集群：
- `cd docker-hadoop`
- `docker-compose up -d`

- 查看 Container 是否已经启动成功：
- `docker ps -a`

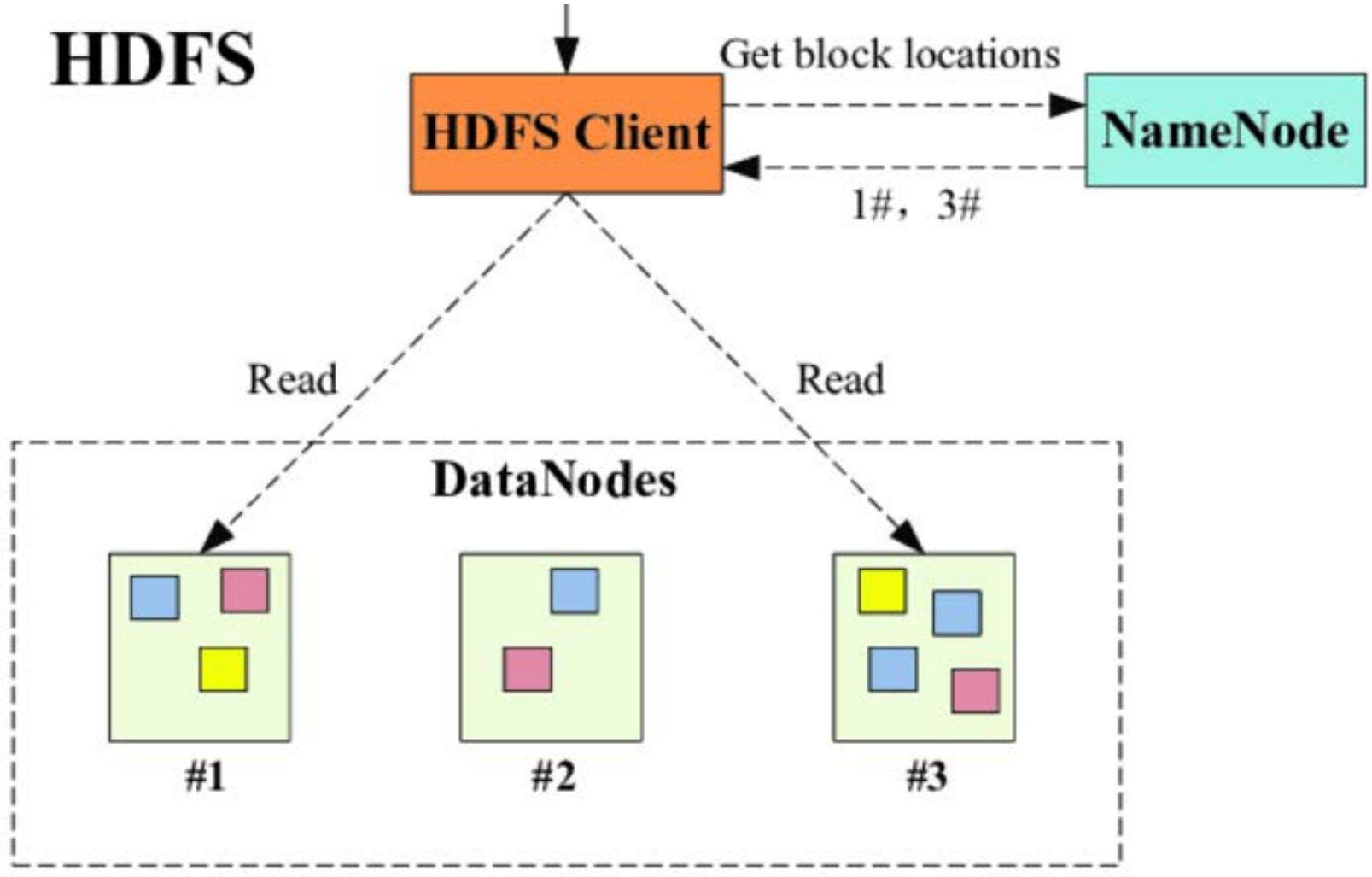
```
~ » docker ps -a
CONTAINER ID        IMAGE                                                COMMAND                  CREATED             STATUS                    PORTS
51702abfe480        bde2020/hadoop-resourcemanager:2.0.0-hadoop3.1.2-java8   "/entrypoint.sh /run…"   8 minutes ago       Up 8 minutes (healthy)    8088/tcp
944cfdc18e87        bde2020/hadoop-datanode:2.0.0-hadoop3.1.2-java8          "/entrypoint.sh /run…"   8 minutes ago       Up 8 minutes (healthy)    9864/tcp
d99465a00550        bde2020/hadoop-nodemanager:2.0.0-hadoop3.1.2-java8       "/entrypoint.sh /run…"   8 minutes ago       Up 8 minutes (healthy)    8042/tcp
8e8ca8ef1b9b        bde2020/hadoop-namenode:2.0.0-hadoop3.1.2-java8          "/entrypoint.sh /run…"   8 minutes ago       Up 8 minutes (healthy)    0.0.0.0:9870->9870/tcp
361015ec0c42        bde2020/hadoop-historyserver:2.0.0-hadoop3.1.2-java8     "/entrypoint.sh /run…"   8 minutes ago       Up 8 minutes (healthy)    8188/tcp
```

安装及使用 HDFS

- 查看 Hadoop 状态：
- http://localhost:9870



- HDFS 的架构
- HDFS 由一个 Name Node 和多个 Data Node
- Name Node 为管理节点
- Data Node 为数据节点



- 测试 HDFS
- 进入 Hadoop 的 Name Node Container:
- `docker exec -it namenode bash`

- 创建两个测试文件:
- `mkdir input`
- `echo "Hello World" >input/f1.txt`
- `echo "Hello Docker" >input/f2.txt`

安装及使用 HDFS

- 在 HDFS 中创建目录：
- `hadoop fs -mkdir -p input`

- 查看 HDFS 的目录和文件：
- `hadoop fs -ls`

```
root@8e8ca8ef1b9b:~# hadoop fs -mkdir -p input
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
root@8e8ca8ef1b9b:~# hadoop fs -ls
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
Found 1 items
drwxr-xr-x   - root supergroup          0 2019-10-14 06:09 input
root@8e8ca8ef1b9b:~#
```

安装及使用 HDFS

- 将文件传入 HDFS:
- hadoop fs -put ./input/* input

```
root@8e8ca8ef1b9b:~# hadoop fs -put ./input/* input
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
root@8e8ca8ef1b9b:~# hadoop fs -ls input
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
Found 2 items
-rw-r--r--   3 root supergroup         12 2019-10-14 06:12 input/f1.txt
-rw-r--r--   3 root supergroup         13 2019-10-14 06:12 input/f2.txt
root@8e8ca8ef1b9b:~#
```

安装及使用 HDFS

- 将文件从 HDFS 复制到本地：
- hadoop fs -get -r input input2

```
root@8e8ca8ef1b9b:~# hadoop fs -get input input2
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
root@8e8ca8ef1b9b:~# ls -al input2
total 16
drwxr-xr-x 2 root root 4096 Oct 14 06:18 .
drwx------ 1 root root 4096 Oct 14 06:18 ..
-rw-r--r-- 1 root root   12 Oct 14 06:18 f1.txt
-rw-r--r-- 1 root root   13 Oct 14 06:18 f2.txt
root@8e8ca8ef1b9b:~#
```

- 将文件从 HDFS 删除：
- hadoop fs -rm -r input

```
root@8e8ca8ef1b9b:~# hadoop fs -rm -r input
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
Deleted input
root@8e8ca8ef1b9b:~# hadoop fs -ls
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
root@8e8ca8ef1b9b:~#
```

安装及使用 HDFS

- 如需将文件传入 Data Node，则需使用：
- hdfs dfs -put ./input/* input

```
root@8e8ca8ef1b9b:~# hdfs dfs -mkdir input
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
root@8e8ca8ef1b9b:~# hdfs dfs -put ./input/* input
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
root@8e8ca8ef1b9b:~# hdfs dfs -ls
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
Found 1 items
drwxr-xr-x   - root supergroup          0 2019-10-14 06:22 input
root@8e8ca8ef1b9b:~#
```

- 更多 Hadoop Docker Container 的使用方法和功能：
- https://clubhouse.io/developer-how-to/how-to-set-up-a-hadoop-cluster-in-docker/

- HDFS Shell 的文档：
- https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html

- Coursera 的 Hadoop 课程
- https://www.coursera.org/learn/hadoop

UC San Diego

目录

Ceph 简介

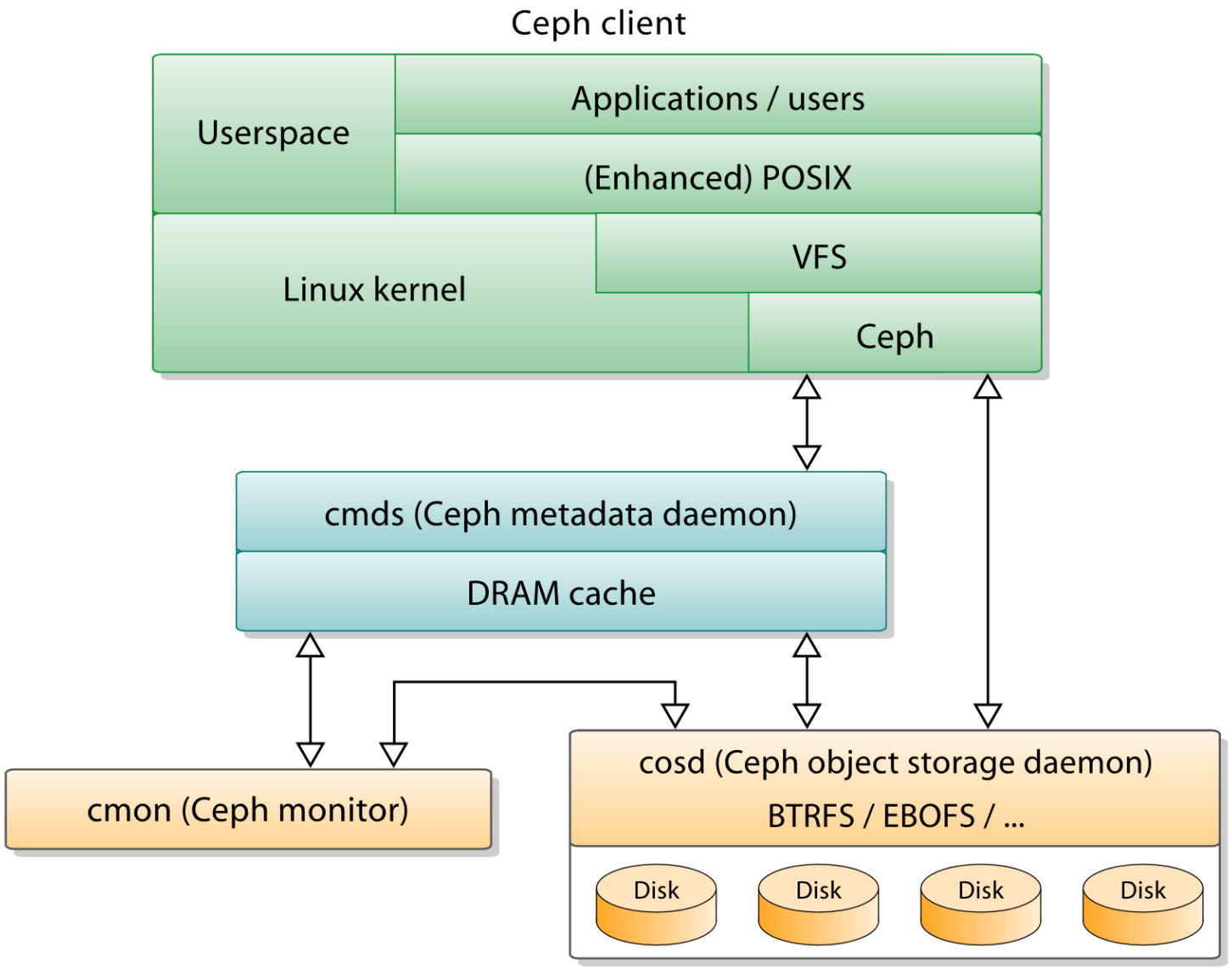
- Ceph
- https://ceph.io/

- Ceph 是一个分布式存储系统，诞生于 2004 年
- 最早致力于开发下一代高性能分布式文件系统的项目
- 随着云计算的发展，Ceph 乘上了 OpenStack 的春风，进而成为了开源社区受关注较高的项目之一

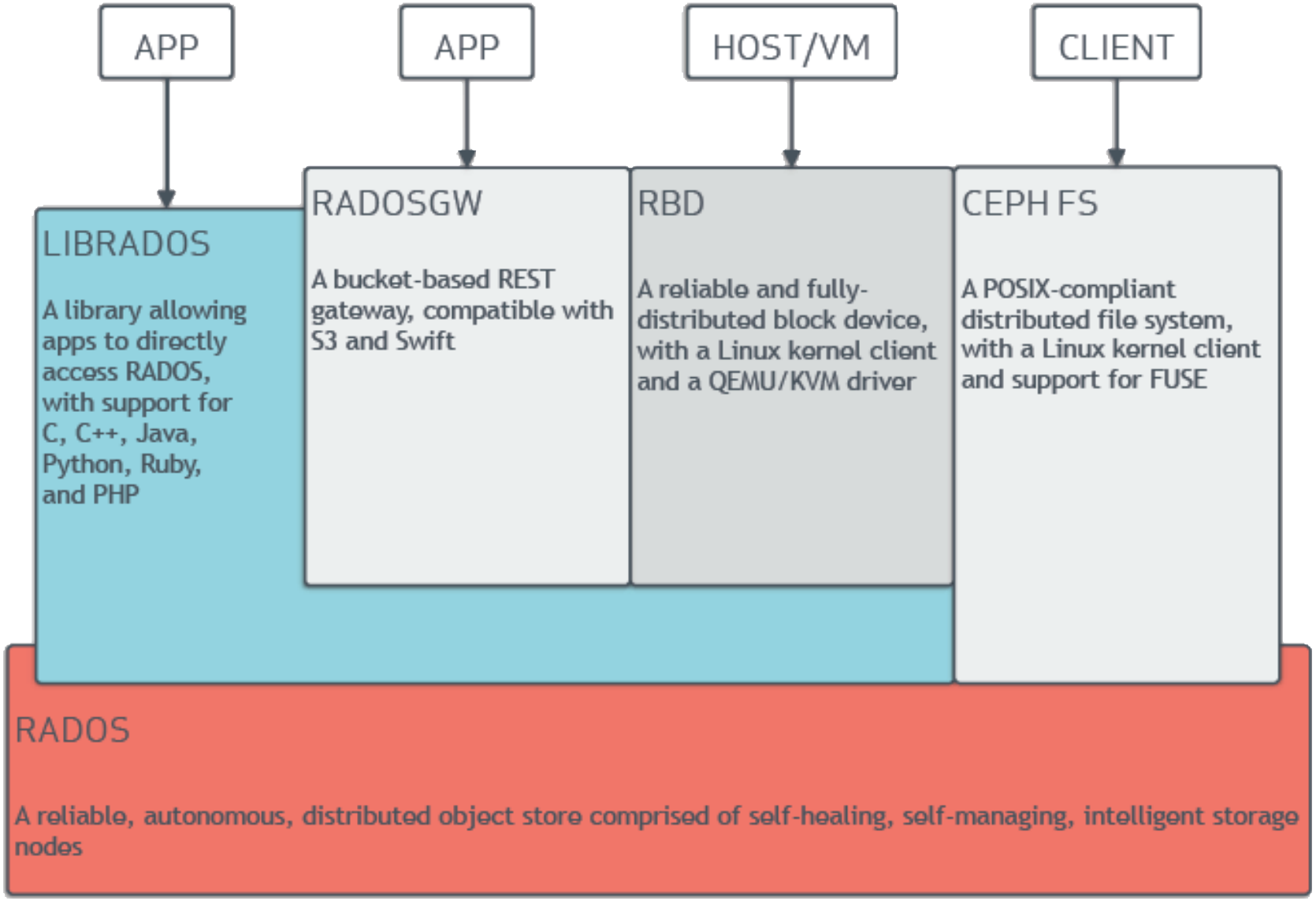
Ceph 简介

- Ceph 的特点
- Crush 算法：Ceph 摒弃了传统的集中式存储元数据寻址的方案，转而使用 Crush 算法完成数据的寻址操作。 Crush 在一致性哈希基础上很好的考虑了容灾域的隔离，能够实现各类负载的副本放置规则，例如跨机房、机架感知等。Crush 算法有相当强大的扩展性，理论上支持数千个存储节点
- 高可用：Ceph 中的数据副本数量可以由管理员自行定义，并可以通过 Crush 算法指定副本的物理存储位置以分隔故障域，支持数据强一致性；Ceph 可以忍受多种故障场景并自动尝试并行修复
- 高扩展性：Ceph 本身并没有主控节点，扩展起来比较容易，并且理论上，它的性能会随着磁盘数量的增加而线性增长
- 特性丰富：Ceph 支持三种调用接口：对象存储，块存储，文件系统挂载。三种方式可以一同使用。在国内一些公司的云环境中，通常会采用 Ceph 作为 OpenStack 的唯一后端存储来提升数据转发效率

❖ Ceph 内部结构


Ceph client
Userspace
Applications / users
(Enhanced) POSIX
Linux kernel
VFS
Ceph
cmds (Ceph metadata daemon)
DRAM cache
cmon (Ceph monitor)
cosd (Ceph object storage daemon)
BTRFS / EBOFS / ...
Disk
Disk
Disk
Disk

- Ceph 核心组件
- Ceph 核心是 RADOS 存储节点，在 RADOS 之上提供以下服务：
- LibRADOS：多语言编程接口
- RADOSGW：对象存储的 REST 接口
- RBD：块存储接口
- CephFS：符合 POSIX 标准的文件系统



安装和使用 Ceph

- Ceph 安装指南
- https://docs.ceph.com/docs/master/

- 安装 Ceph 需要大量 UNIX 知识背景和操作能力
- Ceph 需要接管物理存储设备，硬件需求详情请参考：https://docs.ceph.com/docs/luminous/start/hardware-recommendations/
- 安装和配置 Ceph 不慎可能导致原始数据丢失，因此要格外小心

- Ceph 由 RedHat 社区维护，因此不建议使用 Ubuntu 安装
- 建议使用 CentOS 或 RHEL

安装和使用 Ceph

- Ansible
- https://www.ansible.com/

- Red Hat 社区开发维护的软件自动配置、管理、监控系统
- 由于 Ceph 安装和配置过于复杂，使用 Ansible 可以节约一定时间

安装和使用 Ceph

- 使用 Ansible 安装 Ceph
- https://www.marksei.com/how-to-install-ceph-with-ceph-ansible/

- 安装 Ansible 和 Git:
- `sudo yum install ansible git`

- 配置 /etc/ansible/group_vars/all.yml:

```
ceph_origin: repository
ceph_repository: community
ceph_repository_type: cdn
ceph_stable_release: luminous

monitor_interface: eth0
public_network: 172.16.0.0/16
cluster_network: 10.10.10.0/8
```

安装和使用 Ceph

- 配置 /etc/ansible/group_vars/osds.yml:

```
1.  osd_scenario: non-collocated
2.  osd_objectstore: bluestore
3.  devices:
4.    - /dev/sda
5.    - /dev/sdb
6.  dedicated_devices:
7.    - /dev/sdc
8.    - /dev/sdc
```

- 部署 Ceph:
- cd /usr/share/ceph-ansible
- ansible-playbook -i /path/to/inventory -u $USER site.yml

❖ 如果部署成功，Ansible 会展示 Ceph 的节点详情：

```
PLAY RECAP *********************************************************************************************
mon1                       : ok=180  changed=15   unreachable=0    failed=0
osd1                       : ok=69   changed=5    unreachable=0    failed=0
osd2                       : ok=66   changed=5    unreachable=0    failed=0
osd3                       : ok=66   changed=5    unreachable=0    failed=0
```

安装和使用 Ceph

- 初始化 CephFS:
- `ceph osd pool create cephfs_data <pg_num>`
- `ceph osd pool create cephfs_metadata <pg_num>`

- 在初始化的 Pool 中创建文件系统:
- `ceph fs new cephfs cephfs_metadata cephfs_data`

安装和使用 Ceph

- 确认文件系统创建成功：
- ceph fs ls
- 或：
- ceph fs status

```
$ ceph fs status <cephfs>
cephfs - 1 clients
======
+------+--------+-----------+---------------+-------+-------+
| Rank | State  |    MDS    |    Activity   |  dns  |  inos |
+------+--------+-----------+---------------+-------+-------+
|  0   | active |    mon1   | Reqs:    0 /s |   10  |   12  |
+------+--------+-----------+---------------+-------+-------+
+-----------------+----------+-------+-------+
|       Pool      |   type   |  used | avail |
+-----------------+----------+-------+-------+
| cephfs_metadata | metadata | 2643  | 93.9G |
|   cephfs_data   |   data   |    0  | 93.9G |
+-----------------+----------+-------+-------+

+-------------+
| Standby MDS |
+-------------+
+-------------+
MDS version: ceph version 12.2.11 (26dc3775efc7bb286a1d6d66faee0ba30ea23eee) luminous
 (stable)
```

安装和使用 Ceph

- 挂载文件系统:
- mount -t ceph -o name=<name>,secret=<secret> <mon>:/ <target>

- 或使用 FUSE 挂载:
- sudo ceph-fuse -m <monitor>:<port> <target>

- 更多 Ceph 的使用方法可以参考官方文档：
- https://docs.ceph.com/docs/master/

- CephFS 的安装和使用指南：
- https://www.marksei.com/cephfs-a-beginners-guide/

- Ceph 在阿里云 ECS 中的性能：
- https://yq.aliyun.com/articles/559261
- Ceph 针对大块文件的读写性能非常优秀，高达 2GB/s
- RADOS 读比写高出 10 倍的速率，适合读数据的高并发场景

目录

Alluxio 简介

- Alluxio
- https://www.alluxio.io/

- Alluxio 是一个开源的虚拟分布式文件系统（Virtual Distributed File System，VDFS）
- Alluxio 最初起源于一个叫 Tachyon（快子，一种理论上预测的超光速次原子粒子）的研究项目
- 它是加州大学伯克利分校 AMPLab 实验室由师从 Scott Shenker 教授和 Ion Stoica 教授的李浩源博士的博士论文课题
- Alluxio 位于大数据栈中的计算和存储之间，它为计算框架提供了数据抽象层，使得应用能够通过一个共同的接口连接底层不同的存储系统
- Alluxio 是以 Apache License 的开源协议进行发布的

Alluxio 简介

- UC Berkeley AMPLab
- https://amplab.cs.berkeley.edu/

- 加州大学伯克利分校的一个大数据分析实验室
- AMP = Algorithms、Machines、People
- 主要组织者：Michael J. Franklin、Michael I. Jordan、Ion Stoica
- 主要项目：Apache Mesos、Apache Spark、Alluxio

- 数据编组（Data Orchestration）
- Alluxio 的主要功能和创新
- 将各种不同的云存储方式（包括本地存储）聚合到一起放进内存中或挂载为本地存储
- 屏蔽应用程序访问云存储的具体方式，数据接口完全统一

Inconsistent performance on S3

S3 performance for analytic workloads is inconsistent and data egress is expensive.

Cloud caching solution >

Limited compute capacity on-prem

Making HDFS or object store data accessible to any compute in any cloud is complex.

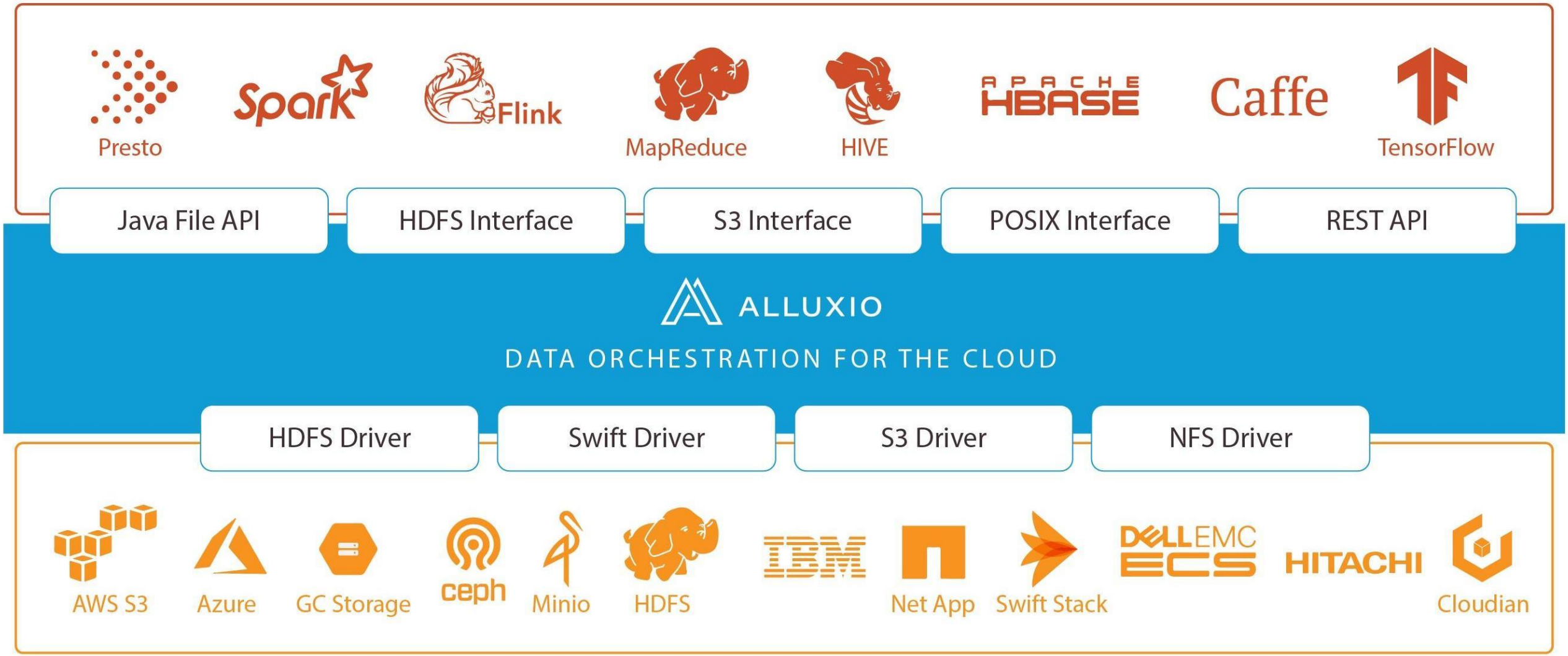
Zero-copy burst solution >

Slow on-prem object store

Object storage performance, particularly for metadata operations, is unpredictable.

Faster workloads on object store solution >

Alluxio 简介


Presto
Spark
Flink
MapReduce
HIVE
APACHE HBASE
Caffe
TensorFlow
Java File API
HDFS Interface
S3 Interface
POSIX Interface
REST API
ALLUXIO
DATA ORCHESTRATION FOR THE CLOUD
HDFS Driver
Swift Driver
S3 Driver
NFS Driver
AWS S3
Azure
GC Storage
ceph
Minio
HDFS
IBM
Net App
Swift Stack
DELL EMC ECS
HITACHI
Cloudian

- 常见的 Alluxio 应用
- Hadoop MapReduce 原本需要访问 HDFS，不支持 Ceph：
- 使用 Alluxio 后可以支持 Ceph，且 MapReduce 并不知情

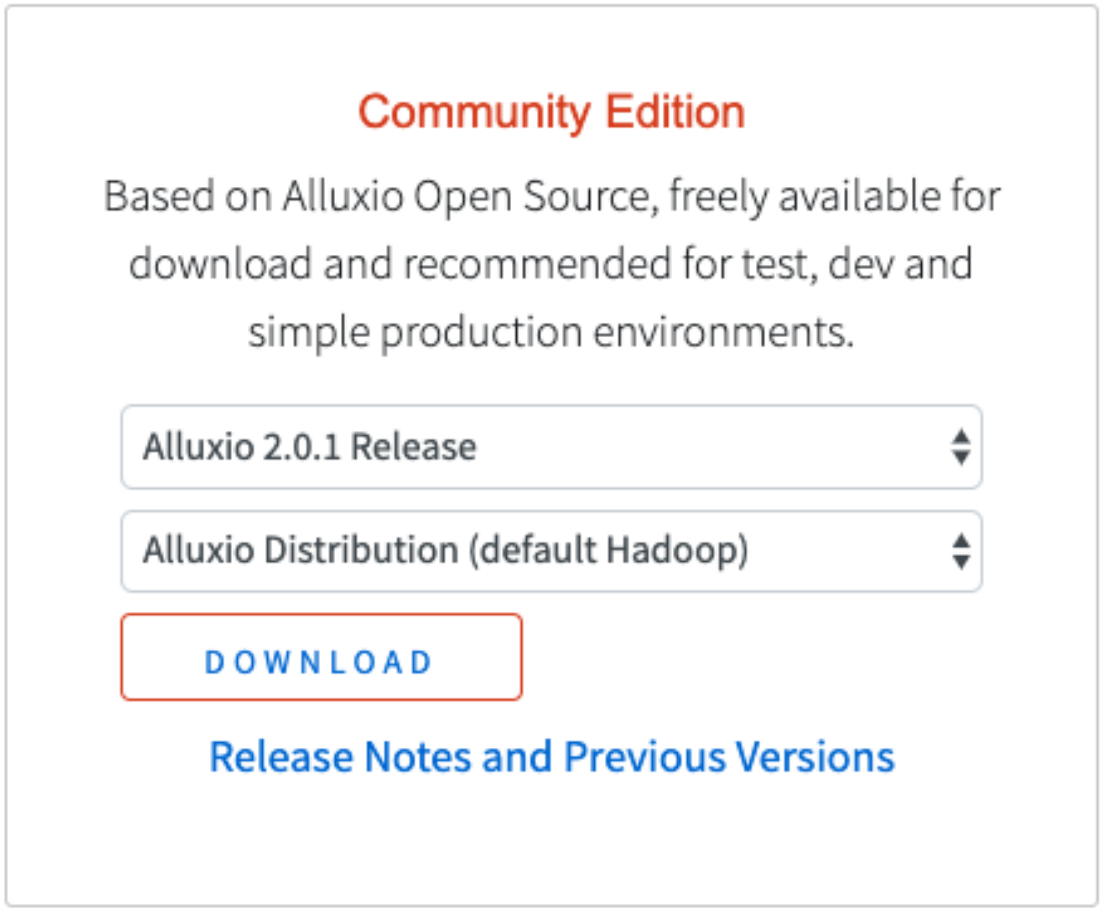
- 将一部分内存虚拟为 POSIX 本地存储，使用数据库（例如 MySQL）访问：
- 可以极大的加快数据库访问速度

安装和使用 Alluxio

- Alluxio 安装指南
- https://www.alluxio.io/download/

- Alluxio 高度支持 AWS S3，可以在 AWS 上直接部署 Alluxio：
- https://aws.amazon.com/marketplace/seller-profile?id=c95fb274-91ef-4d5c-bf24-d7312c282d6e

- 本地部署只需要下载、安装、运行即可



安装和使用 Alluxio

- Alluxio + Presto Docker Container
- https://www.alluxio.io/alluxio-presto-sandbox-docker/

- `docker pull alluxio/alluxio-presto-sandbox`
- `docker run -d --shm-size 1G -p 19999:19999 -p 8080:8080 alluxio/alluxio-presto-sandbox`

- 端口号：
- 19999 为 Alluxio 的 Web UI
- 8080 为 Presto 的 Web UI

- Presto
- https://prestodb.github.io/

- Presto 是一种用于大数据的高性能分布式 SQL 查询引擎
- 其架构允许用户查询各种数据源，如 Hadoop、AWS S3、Alluxio、MySQL、Cassandra、Kafka 和 MongoDB
- Presto 甚至可以在单个查询中查询来自多个数据源的数据
- Presto 是 Apache许可证下发布的社区驱动的开源软件

```
$ presto
presto:default> describe nation;
   Column     |  Type   | Null | Partition Key
--------------+---------+------+---------------
 n_nationkey  | bigint  | true | false
 n_name       | varchar | true | false
 n_regionkey  | bigint  | true | false
 n_comment    | varchar | true | false
(4 rows)

Query 20131105_005529_00080_ee7y3, FINISHED, 2 nodes
Splits: 2 total, 2 done (100.00%)
0:00 [8 rows, 446B] [23 rows/s, 1.29KB/s]

presto:default>
```

安装和使用 Alluxio

- 连接进入 Alluxio + Presto 的 Container：
- `docker exec -it alluxio-presto-sandbox bash`

- 查看已经挂载的文件系统：
- `alluxio fs mount`

- 在这个 Container 里，Alluxio 已经预先挂载了一个 AWS S3 的对象存储目录，因此执行上述命令后应当得到如下结果：

```
s3://alluxio-public-http-ufs/tpcds/scale1-parquet  on  /scale1  (s3, capacity=-1B, used=-1B, read-only, not shared, proper
/opt/alluxio/underFSStorage                        on  /        (local, capacity=58.42GB, used=-1B(0%), not read-only, not
```

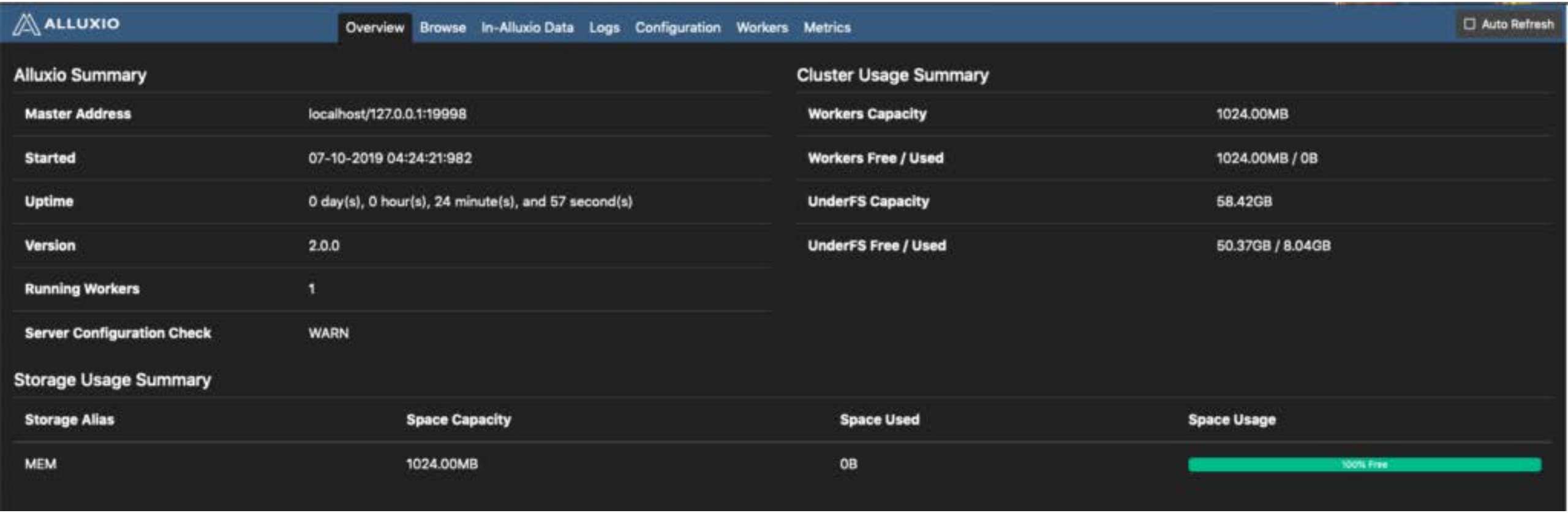
安装和使用 Alluxio

- 查看 Alluxio 文件系统的内容：
- `alluxio fs ls /scale1`

```
drwx------                3       PERSISTED 07-05-2019 19:41:42:054  DIR
/scale1/call_center
drwx------                3       PERSISTED 07-05-2019 19:41:42:100  DIR
/scale1/catalog_page
drwx------             2067       PERSISTED 07-05-2019 19:41:52:535  DIR
/scale1/catalog_returns
drwx------             1832       PERSISTED 07-05-2019 19:41:59:149  DIR
/scale1/catalog_sales
drwx------                3       PERSISTED 07-05-2019 19:41:59:152  DIR /scale1/customer
drwx------                3       PERSISTED 07-05-2019 19:41:59:156  DIR
/scale1/customer_address
drwx------                3       PERSISTED 07-05-2019 19:41:59:161  DIR
/scale1/customer_demographics
drwx------                3       PERSISTED 07-05-2019 19:41:59:165  DIR /scale1/date_dim
drwx------                3       PERSISTED 07-05-2019 19:41:59:169  DIR
/scale1/household_demographics
drwx------                3       PERSISTED 07-05-2019 19:41:59:173  DIR
/scale1/income_band
drwx------              262       PERSISTED 07-05-2019 19:42:00:104  DIR /scale1/inventory
```

安装和使用 Alluxio

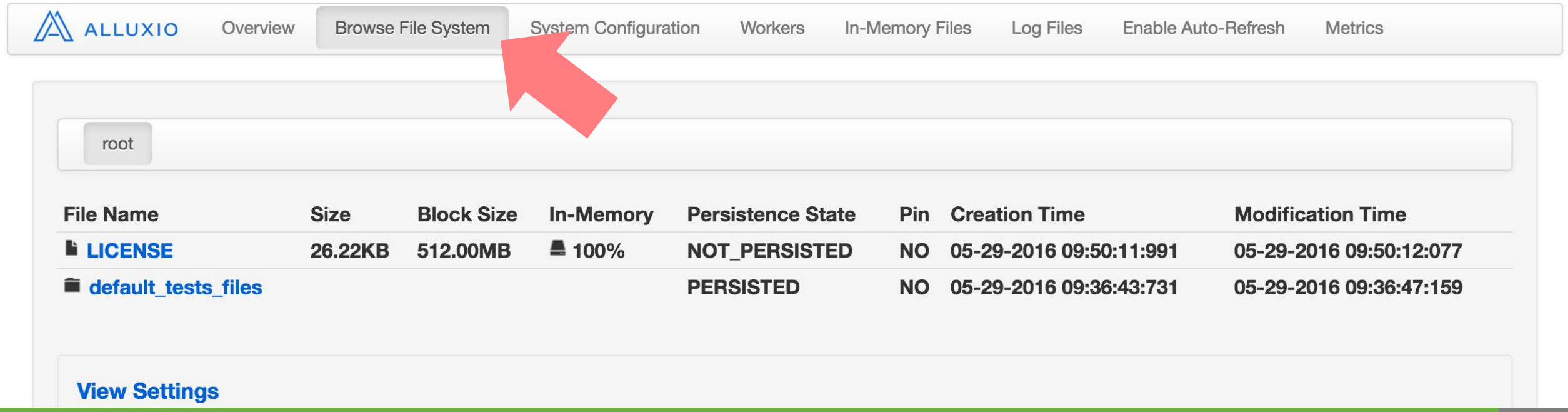
- 查看 Alluxio 的 Web UI：
- http://localhost:19999/



安装和使用 Alluxio

- 从本地加载数据到 Alluxio 中：
- `alluxio fs load /scale1/customer_demographics`

```
/scale1/customer_demographics/part-00000-68449736-ad44-43d2-841f-4d55afd9e0b3-c000.snappy.parquet loaded
/scale1/customer_demographics/part-00000-267c2412-d427-4907-a398-e6de535ff1d4-c000.snappy.parquet loaded
/scale1/customer_demographics/_SUCCESS already in Alluxio fully
/scale1/customer_demographics loaded
```



- Alluxio 的 CLI 文档：
- https://docs.alluxio.io/os/user/1.7/en/Command-Line-Interface.html

- Alluxio 在携程大数据平台的实践：
- https://www.bilibili.com/video/av38523137

- Data Orchestration for Analytics and AI in the Cloud：
- https://www.alluxio.io/resources/presentations/alluxio-data-orchestration-for-analytics-and-ai-in-the-cloud/

目录

IPFS（星际文件系统）

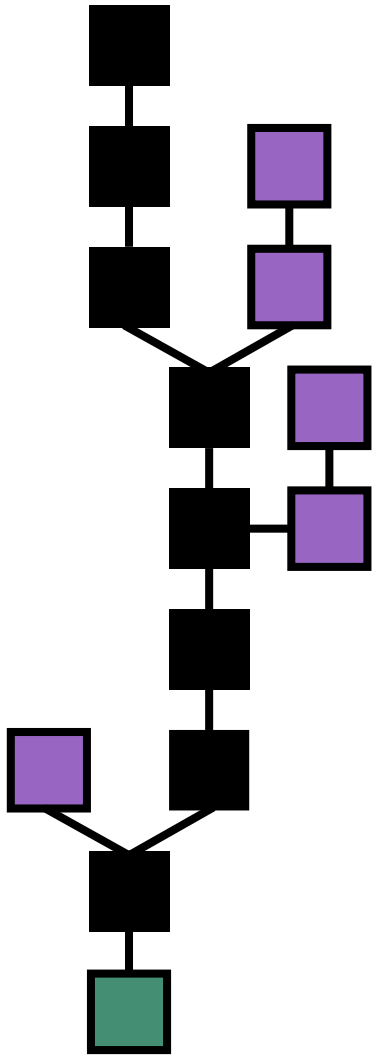
IPFS 简介

- InterPlanetary File System（IPFS）
- https://ipfs.io/

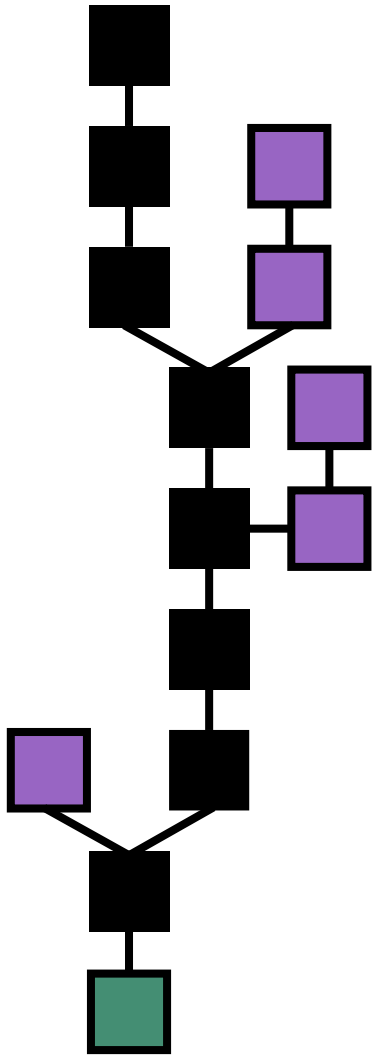
- 基于区块链的文件托管服务，于 2014 年首次提出，使用 Go 实现
- IPFS 协议利用比特币区块链协议和网络基础设施的优势来存储不可更改的数据
- IPFS 提供了一个高吞吐量、按内容寻址的块存储模型
- 简单来讲就是用区块链来实现去中心化的私有云
- 由于区块链的不可更改性和易于发行数字货币的特性，IPFS 更容易对私有云收费

IPFS 简介

- 区块链（Blockchain）
- 一种速度极慢、安全性极高的数据库
- 区块链由多个区块（Block）组成，每个区块包含了一定的数据
- 除此之外，每一个区块还包含了前一个区块的加密散列（Hash）、相应时间戳记等额外信息
- 由于每个区块的加密散列都包含前一区块数据，多个区块形成一条链
- 修改任何一个历史区块都需要从被修改区块开始更新全部的加密散列
- 这样的设计使得区块内容具有难以篡改的特性
- 用区块链技术所串接的账本能让两方有效纪录交易，且可永久查验此交易



- 共识（Consensus）
- 一条记录必须由多个记录者确定（Confirm）并达成共识后，方可生效（成为不可篡改的永久记录）
- 单个记录者篡改记录不一定会获得其他记录者的同意
- 达成共识的记录者越多，共识被打破的概率越低
- 由一部分记录者打破共识，并形成有别于原始记录的另一条区块链的行为被称为：分叉（Branch Off）



IPFS 简介

- 比特币（Bitcoin）
- 区块链 + 共识机制的典型应用
- 比特币为一个分布式账本，其中记录了用户比特币转账的信息
- 比特币还引入了奖励机制，要求记录者记录信息的用户需要为记录者转移一定的比特币，记录者才会为用户记录信息
- 针对奖励机制，比特币还引入了竞争机制，即记录者可以争相为用户记录数据，但必须通过一个毫无意义，但具有竞争价值的机制竞争：计算散列函数
- 这种竞争机制称为工作量证明（Proof-of-Work，PoW）
- 如果将比特币形容为“金矿”，则替用户记录数据并索取报酬被称为“挖矿”
- 由于区块链 + 共识机制的账本具有极高的安全性，比特币等类似的区块链应用常被称为数字货币（Cryptocurrency）

IPFS（星际文件系统）

IPFS 简介

- IPFS 的特点
- IPFS 也是基于区块链 + 共识机制的应用，因此也具备以下特点：
- IPFS 也具备极高的安全性和极慢的访问速度
- IPFS 也有奖励机制，因此也会记录转账信息
- 由于 IPFS 会记录转账信息，IPFS 也会产生数字货币
- IPFS 也有设立竞争机制用于公平对待所有的记录者

IPFS（星际文件系统）

IPFS 简介

- IPFS 的特点
- 不同于传统的区块链应用，IPFS 具备以下的新特点：
- IPFS 不仅记录转账信息，也记录文件或文件的位置，所以属于云存储的一种
- IPFS 的奖励机制为权益证明（Proof-of-Stake），并非工作量证明

❖ IPFS 的工作流程

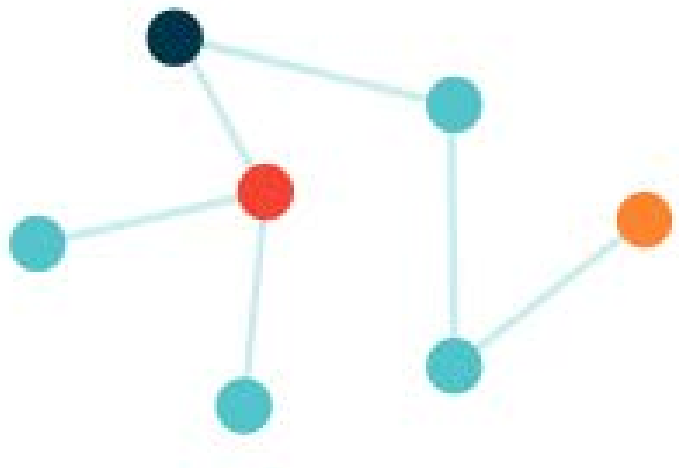
❖ 首先，IPFS 会对一个文件的所有分块建立全局唯一 ID

Your file, and all of the **blocks within it**, is given a **unique fingerprint** called a **cryptographic hash.**

- IPFS 的工作流程
- 其次，IPFS 会对全部文件分块去冗余

IPFS **removes duplications** across the network.

- IPFS 的工作流程
- 接下来，IPFS 会将文件写入区块链，并存放于共识网络中
- 不同的共识网络节点不一定都会存放文件的全部内容，也可能仅存放一份索引



Each **network node** stores only content it is interested in, plus some indexing information that helps figure out which node is storing what.

❖ IPFS 的工作流程

❖ 当用户用需要查看某个文件时，共识网络节点会为用户传输文件，或告知用户存放有所需文件的节点的地址

When you **look up a file** to view or download, you're asking the network to find the nodes that are storing the content behind that file's hash.

- IPFS 的工作流程
- 为了方便用户搜索文件，IPFS 还设立了 IPNS 服务器来解析文件名

You don't need to remember the hash, though — every file can be found by **human-readable names** using a decentralized naming system called **IPNS**.

IPFS（星际文件系统）

IPFS 简介

- IPFS 的工作流程
- 整体来讲，IPFS 类似分布式网络传输协议
- 但是，相对于传统的分布式网络传输协议，IPFS 具备以下特点：
- 便捷收费：共识节点可以通过提供文件服务快速收费，而且用户难以破解
- 安全：由于基于区块链技术，IPFS 的文件难以篡改
- 完全去中心化：由于采用共识机制，IPFS 完全没有中心服务器

IPFS（星际文件系统）

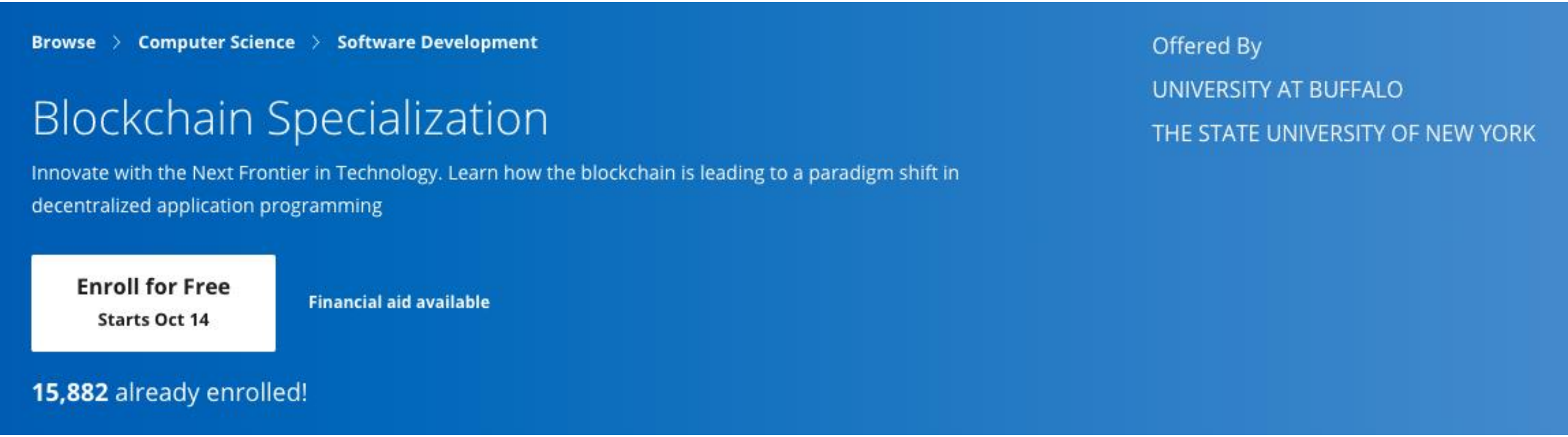
IPFS 进阶

- IPFS 白皮书：
- https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf

- IPFS 文档：https://docs.ipfs.io/

- IPFS 详细部署流程：
- https://www.youtube.com/watch?v=8CMxDNuuAiQ

- Cousera 的区块链课程：
- https://www.coursera.org/specializations/blockchain



目录

其他常见的分布式云存储系统

Lustre

- Lustre
- http://lustre.org/

- Lustre 是一种平行分布式文件系统，通常用于大型计算机集群和超级计算机
- Lustre 是源自 Linux 和 Cluster 的混成词
- 最早在 1999 年由 Peter Braam 创建的集群文件系统公司（Cluster File Systems Inc.）开始研发，于 2003 年发布 Lustre 1.0
- 采用 GNU GPLv2 开源码授权

其他常见的分布式云存储系统

Gluster

- Gluster
- https://www.gluster.org/

- 原名 GlusterFS
- 类似 Lustre，Gluster 也是一种平行分布式文件系统
- Gluster 是源自 GNU 和 Cluster 的混成词
- Gluster 和 Lustre 没有任何关系
- Gluster 和 Ceph 一样都由 Red Hat 社区维护

总结

- 三大云服务平台
- Google Cloud：https://cloud.google.com
- Amazon Web Services：https://aws.amazon.com
- 阿里云：https://www.aliyun.com

总结

- 课外阅读
- 《云存储技术——分析与实践》，刘洋著，经济管理出版社
- http://product.dangdang.com/24247525.html

- 《Ahead in the Cloud》，Stephen Orban（GM of AWS）
- https://www.amazon.com/Ahead-Cloud-Practices-Navigating-Enterprise/dp/1981924310/

- 《Cloud Computing: Concepts, Technology & Architecture》，Thomas Erl
- https://www.amazon.com/Cloud-Computing-Concepts-Technology-Architecture/dp/0133387526/

Thanks!