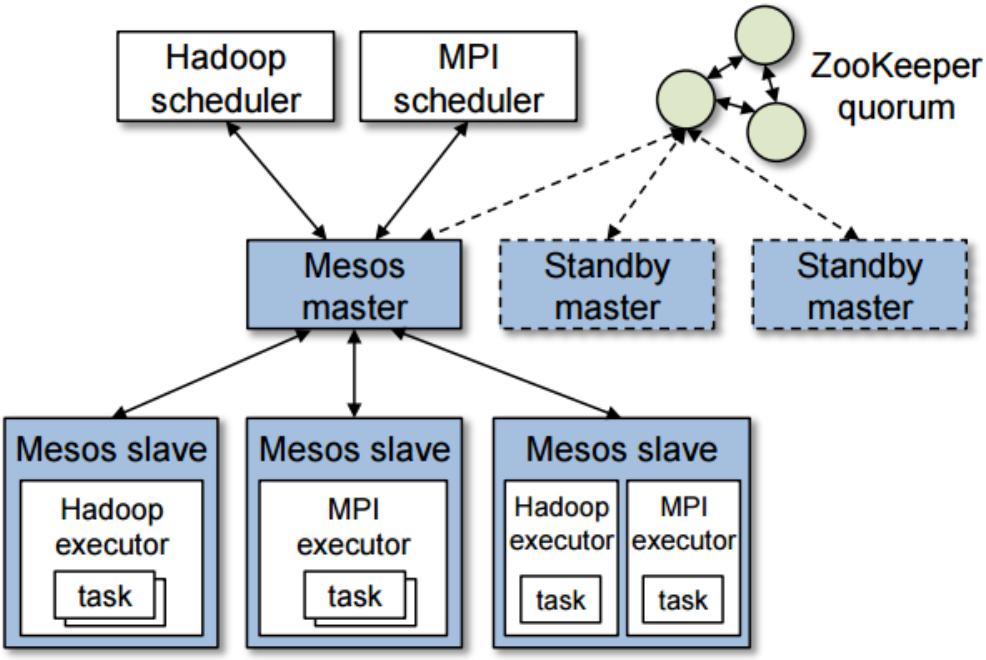
作业一

学号：1300013022

姓名：武守北

一、阅读 Mesos 论文《Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center》，并了解数据中心操作系统的概念：

1、论文《Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center》提出了 Mesos，主要内容： Mesos 是一个支持在多种计算集群框架（frameworks）间共享服务器集群的平台，利用 HADOOP，MPI，提高了集群资源占用率，避免了每 种框架的数据重复。Mesos 能够镜像细粒度的资源共享，通过轮流的读取磁盘数据是的 frameworks 能从本地获取数据。为了满足复杂的资源调度方法，Mesos 引入了称为资源提供的（resource offer）的 2 层资源调度机制。Mesos 决定多少资源分配给 frameworks，frameworks 决定接受多少资源和决定哪个任务使用多少资源。 Mesos，作为一个薄的资源共享层，通过对集群框架提供共有的访问集群资源的接口，使得在多样化的集群计算框架中实现细粒度共享成为可能 数据中心操作系统主要做到了将数据中心的大规模服务器集群视为了一个计算机，对其中的 CPU、内存、储存装置以及其他运算资源，全部加以虚拟化，并进行管理。同时，如 Spark 这类并行计算框架或 Hadoop 这类分布存储框架等应用都可以运行于其上。Mesos 的具体架构如下：



具体的工作流程：

如上图所示，Mesos 的架构有 ZooKeeper、Mesos Master、Standby Master、Mesos slaves 以及 Framework。其中 ZooKeeper 与 Standby Master 不是必须的。 MPI 和 Hadoop 是两个 Framework 的例子。Task 通过 Framework 来运行。 Master 调配资源，Slave 运行 Master 分配给它的 Task，如上图所示一个 Slave 也可以运行两个 Tasks。

Master 负责管理 Slave 在每个节点上的运行， 根据指定的策略决定提供多少的资源给 Framework。

Slave（Agent）定期向 Master 汇报拥有的资源，然后根据 Master 的调配执行对应的命令， 以及管理分配给自己的任务。

Framework 包括 scheduler 和 executor 两部分。 scheduler 注册到 Master 以获取集群资源，并在获取资源后，选择其中提供的一些资源，然后将 tasks 发送到 slave 上进行运行。executor 运行在 slave 上，负责执行 framework 的 tasks。

Standby Master 是备用的 Master。Master 的设计是 soft state 的，可以通过从 slave 和 framework 获得的消息完全恢复状态并工作。当原有的 Master 发生错误时，为了保证运行在其上的 framework 可以继续工作，需要一个新的 Master。

ZooKeeper 在重新选举一个 Master 起作用。另外，当 Agent 检测 Master 时，如果指定了 ZooKeeper，则会使用 ZooKeeper 做 Master 的检测，不然则使用 StandaloneMasterDetector。抽象一下，就是：

1、Slave 定期向 Master 报告自身可用的资源。

2、Master 发送可用资源给 Framework，让 Framework 进行调配。

3、Framework 答复 Master 具体在哪个 Slave 上运行什么 Task，Task 需要多少资源。

4、Master 将 Tasks 发送给 Slaves，然后 Slaves 运行 Task，报告可用资源给 Master。

二、了解虚拟机和容器技术，用自己的话简单叙述、总结并对比：

虚拟机：一般来讲，虚拟机指通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统。虚拟机是一个用软件模拟出来的物理计算机，为上层的提供的是指令集（ISA）和 IO 硬件层次的接口。

容器：容器为用户程序提供了一个隔离的运行环境，容器内的程序不会影响到其他容器和宿主机的内容，为上层提供的是一个操作系统和运行环境。容器和宿主机共享一个操作系统内核，用宿主机操作系统负责调度系统资源。

二者都是一种虚拟化技术，都可以为用户提供一个隔离的运行环境。相比而言，虚拟机提供了更底层的虚拟化，支持对不同的操作系统，甚至不同的体系结构的虚拟化。而容器则更加侧重于提供一个与宿主机相似且隔离的运行环境。容器的虚拟化程度和隔离程度都不如虚拟机高，因此也能获得体积和性能上的优势。虚拟机是通过操作系统隔离的，开销较大，是重量级的，分钟级启动的；而容器是通过进程技术隔离的，开销较小，是轻量级的。而比容器虚拟机相对于容器更加安全，因为虚拟机提供了专用的操作系统和更牢固的逻辑边界。而彼此相邻的容器共享处理器。两种技术在使用场景上的差距也是非常大的， 虚拟机的花我们平常用在从原有的系统中分离去一定的计算机硬件资源去运行另一个系统级的应用。而容器的花很多容器相当于在系统上运行的一个程序没，而这个程序帮我们管理我们的其他程序。所以说容器和虚拟机仅仅相似于它们都提供了隔离环境。容器比起虚拟机能做的事少得多并且使用起来相当廉价。而虚拟机提供整个虚拟化硬件层，可以做更多的事情但是使用成本显著。

三、从 github 上获取 mesos 项目，切换到 tag 为 1.1.0 的版本自己 build 并运行起来：

下载 mesos 1.1.0 并运行：

```
# Change working directory.
$ cd mesos

# Bootstrap (Only required if building from git repository).
```

```
$ ./bootstrap

# Configure and build.
$ mkdir build
$ cd build
$ ../configure
$ make

# Run test suite.
$ make check

# Install (Optional).
$ make install
```

四、运行 Spark on Mesos，以不同并行度运行两次 wordcount 程序并比较，需要在报告中详细说明并附资源使用情况及时间花费截图：

下载 Spark_2. 1. 0，解压、设置配置文件、压缩并且放到对应路径下；

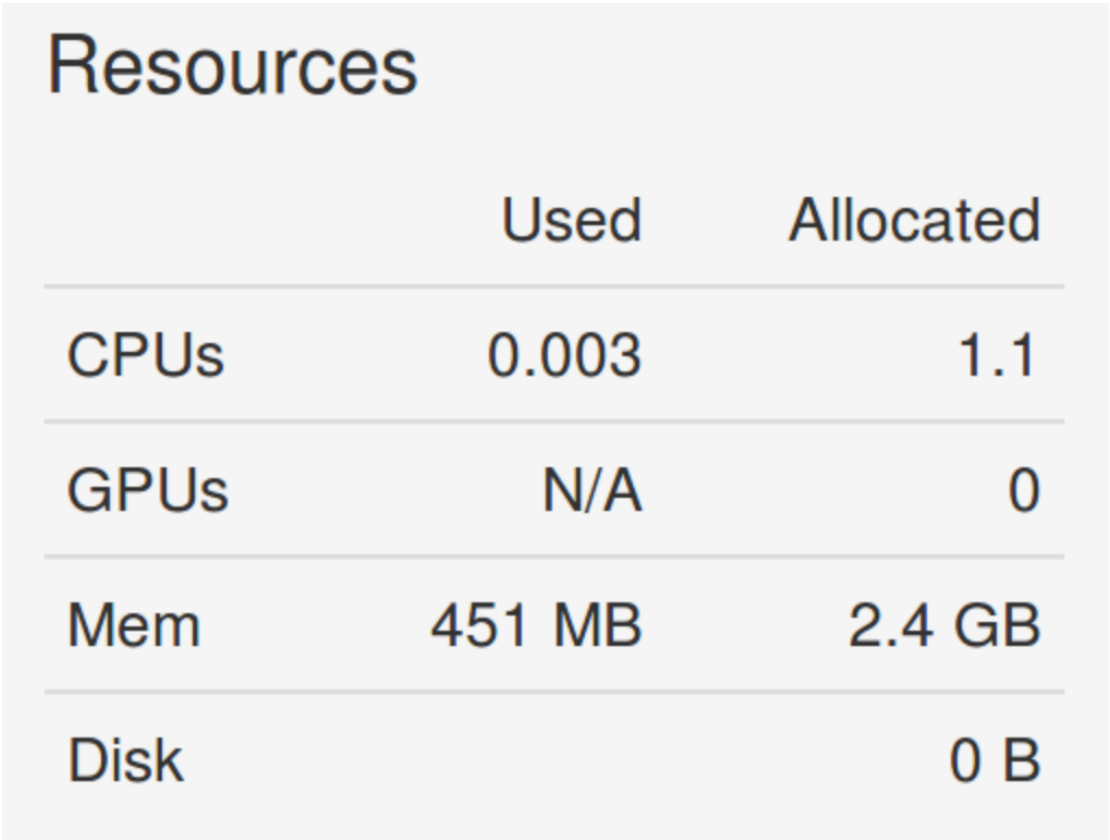
使用的 test.txt 文件大小为 200M；

1、CPU 个数为 1，Memory 为 2G 运行 wordcount 程序：

Resources

	Used	Allocated
CPUs	0.010	1.1
GPUs	N/A	0
Mem	1.4 GB	2.4 GB
Disk		0 B

2、CPU 个数为 2，Memory 为 2G 运行 wordcount 程序：



CPU 核数增加后，运行时间显著减少，内存使用也大幅下降。但我们需要结合 test.txt 的具体大小去做判断。

五、叙述自己对这些软件技术与具体安装运行过程的看法：

通过安装，出现问题与解决问题，对 mesos 与 spark 的整个框架有了初步的了解。运行在 mesos 上面和 spark standalone 模式的区别是：

1、stand alone：

需要自己启动 spark master

需要自己启动 spark slaver（即工作的 worker）

2、运行在 mesos：

启动 mesos master

启动 mesos slaver

```
./sbin/start-mesos-dispatcher.sh -m mesos://127.0.0.1:5050
```

启动 spark

配置 spark 的可执行程序的路径(也就是 mesos 里面所谓 EXECUTOR),提供给 mesos 下载运行。

在 mesos 上面的运行流程：

1、通过 spark-submit 提交任务到 spark-mesos-dispatcher

2、spark-mesos-dispatcher 把通过 driver 提交到 mesos master，并收到任务 ID

3、mesos master 分配到 slaver 让它执行任务

4、spark-mesos-dispatcher，通过任务 ID 查询任务状态