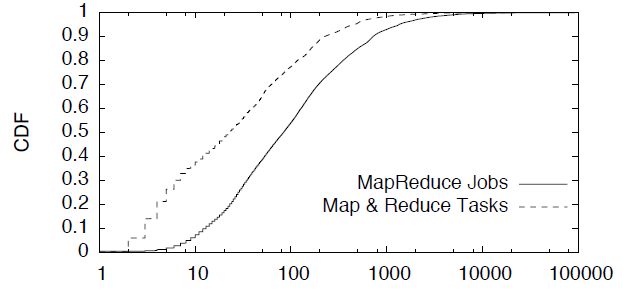
Apache Mesos使用及性能测试

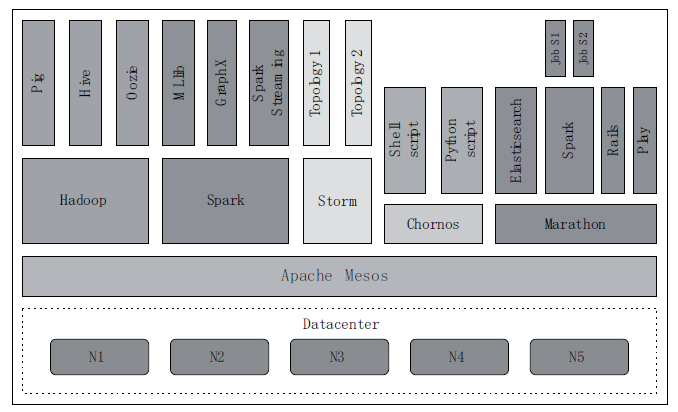
# 一、简介

Apache Mesos是分布式集群管理框架，最初由AMPLab开发，其起源于Google的数据中心资源管理系统Brog，在2010年成为Apache项目。Mesos使用的一个典型场景是Facebook的业务，Facebook内有Hadoop data warehouse，将web Services的日志加载到1200个节点的Hadoop Cluster中，用于商业智能，垃圾邮件检测及Ad优化。另外一些运营job周期执行，这些Job从几个小时的机器学习到1-2分钟的Ad-hoc会话，如图1所示：



为了满足这些Job的需求，Facebook使用fair-sheduler作为调度决策，这个调度策略重复考虑了Map/Reduce Tasks，数据本地性等细粒度的特性。但是不好的地方是，Cluter仅能运行Hadoop 任务，如果开发者想在集群中运行MPI(job之间交互的效率高)，那么需要建立单独的MPI Cluster，然后将数TB的数据再导入该集群，代价太大。Mesos旨在解决同一集群多种Framework的执行，而且Framewokr对资源有足够的控制，可以满足数据本地性这样的放置目的。

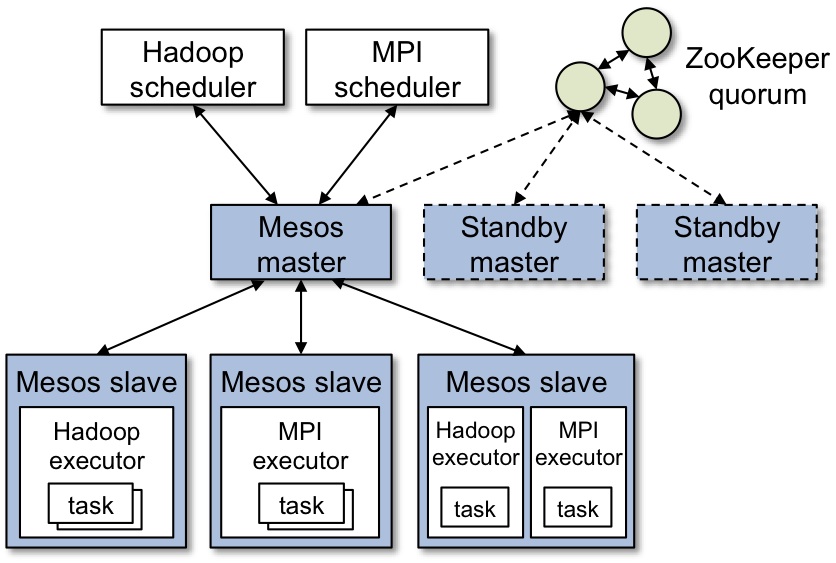
Mesos处于操作系统和应用层之间，提供有效的资源隔离和共享，使用Cgroup完成CPU、内存、I/O、文件系统等资源的隔离；Mesos在多种不同类型的Framework之间共享集群资源，如下图所示，Mesos可以看做是数据中心的内核：



Apache Mesos使应用运行在大规模集群资源共享池中，类似于内核在单台机器上的作用，使应用可以无缝的访问集群资源。Mesos的核心是两级调度架构，Mesos决定Framework的资源使用量，Framework完成资源的分配和程序的运行。目前Mesos支持的Framework包括Hadoop、Memcached、Ruby on Rails、Strom、JBoss Data Grid、MPI、Spark及Node.js等，也包括多种Web Servers、数据库及应用服务器等。

# 二、系统架构

Mesos框架如下图所示：

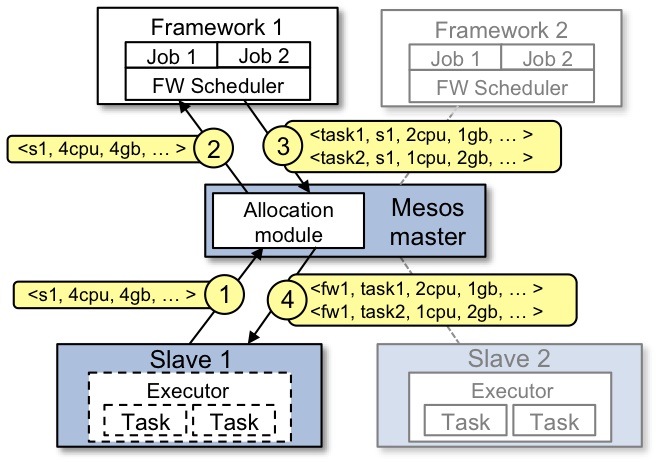


Mesos的组件包括Master，Slave及Mesos Applications。其中Slave运行在每个节点上，App运行在Slave上，Master管理所有的Slave。

Master为Framework提供cpu,ram等资源，每中资源都是一个资源列表。Master按照一定的策略，例如fair sharing，strict priority，为Framework分配资源。

运行在Mesos上的框架，包括两个Components：scheduler（注册到Master中，用于调度资源），executor用于向Slave nodes上指派tasks。Master决定为App提供多少资源，frameworks的scheduler决定如何使用资源。当一个Frameworks收到提供的资源，它会将要运行的Tasks的描述发送给Master。Mesos在相应的Slaves上发布Task。

下面是一个资源分配及Task调度的例子，如下图所示：



运行步骤如下所示：

1. Slave1向Master汇报资源情况，<4Cpu,4GB>可被使用。Master触发Allocation policy Module，Module告诉Framework1,可提供资源
2. Master向framework发送资源offer（描述Slave1上可使用的资源）
3. Framework Scheduler向master响应，告诉master在该Slave上可运行2个Tasks，Task1: <2Cpus,1GB>,Task2: <1Cpu,2GB>
4. 最后，Master将Tasks启动请求，发送到slave1上，Slave1启动这两个Tasks。但是由于还有<1Cpu,1GB>没被使用，因此Master会将剩余资源分配给Framework 2。

资源offer process会在task完成或新资源出现时重复调用。

# 三、安装

## 3.1安装环境依赖

# Install a recent kernel for full support of process isolation.

$ sudo rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org

$ sudo rpm -Uvh http://www.elrepo.org/elrepo-release-6-6.el6.elrepo.noarch.rpm

$ sudo yum --enablerepo=elrepo-kernel install -y kernel-lt

# Make the just installed kernel the one booted by default, and reboot.

$ sudo sed -i 's/default=1/default=0/g' /boot/grub/grub.conf

$ sudo reboot

# Install a few utility tools. This also forces an update of `nss`,

# which is necessary for the Java bindings to build properly.

$ sudo yum install -y tar wget git which nss

# 'Mesos > 0.21.0' requires a C++ compiler with full C++11 support,

# (e.g. GCC > 4.8) which is available via 'devtoolset-2'.

# Fetch the Scientific Linux CERN devtoolset repo file.

$ sudo wget -O /etc/yum.repos.d/slc6-devtoolset.repo http://linuxsoft.cern.ch/cern/devtoolset/slc6-devtoolset.repo

# Import the CERN GPG key.

$ sudo rpm --import http://ftp.riken.jp/Linux/cern/centos/7.1/os/x86\_64/RPM-GPG-KEY-cern

# Fetch the Apache Maven repo file.

$ sudo wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo

# 'Mesos > 0.21.0' requires 'subversion > 1.8' devel package, which is

# not available in the default repositories.

# Create a WANdisco SVN repo file to install the correct version:

$ sudo cat > /etc/yum.repos.d/wandisco-svn.repo <<EOF

[WANdiscoSVN]

name=WANdisco SVN Repo 1.8

enabled=1

baseurl=http://opensource.wandisco.com/centos/6/svn-1.8/RPMS/$basearch/

gpgcheck=1

gpgkey=http://opensource.wandisco.com/RPM-GPG-KEY-WANdisco

EOF

# Install essential development tools.

$ sudo yum groupinstall -y "Development Tools"

# Install 'devtoolset-2-toolchain' which includes GCC 4.8.2 and related packages.

$ sudo yum install -y devtoolset-2-toolchain

# Install other Mesos dependencies.

$ sudo yum install -y apache-maven python-devel java-1.7.0-openjdk-devel zlib-devel libcurl-devel openssl-devel cyrus-sasl-devel cyrus-sasl-md5 apr-devel subversion-devel apr-util-devel

# Enter a shell with 'devtoolset-2' enabled.

$ scl enable devtoolset-2 bash

$ g++ --version  # Make sure you've got GCC > 4.8!

# Process isolation is using cgroups that are managed by 'cgconfig'.

# The 'cgconfig' service is not started by default on CentOS 6.6.

# Also the default configuration does not attach the 'perf\_event' subsystem.

# To do this, add 'perf\_event = /cgroup/perf\_event;' to the entries in '/etc/cgconfig.conf'.

$ sudo yum install -y libcgroup

$ sudo service cgconfig start

## 3.2编译及安装

1.下载mesos并编译

$wget http://www.apache.org/dist/mesos/0.28.0/ [mesos-0.28.0.tar.gz](http://www.apache.org/dist/mesos/0.28.0/mesos-0.28.0.tar.gz)

$tar -zxf [mesos-0.28.0.tar.gz](http://www.apache.org/dist/mesos/0.28.0/mesos-0.28.0.tar.gz)

2.编译并install

# 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

3.配置mesos环境

1）/usr/local/etc/mesos目录下创建并配置相关文件。

创建masters和slaves文件

|  |
| --- |
| $ cat masters  fys1.cmss.com  $ cat slaves  fys1.cmss.com  fys2.cmss.com  fys3.cmss.com |

2）配置mesos-master-env.sh：

|  |
| --- |
| export MESOS\_log\_dir=/data/mesos/master/log  export MESOS\_work\_dir=data/mesos/master/work  export MESOS\_ZK=zk://fys1:2181/mesos  export MESOS\_quorum=1 |

3）配置mesos-slave-env.sh：

|  |
| --- |
| export MESOS\_master=bdi13.cmss.com:5050  export MESOS\_log\_dir=/data/mesos/slave/log  export MESOS\_work\_dir=/data data/mesos/slave/work  export MESOS\_isolation=cgroups |

4.mesos目录结构

|—/usr/

|—local/libexec/mesos/

|—mesos-executor

|—mesos-docker-executor

|—mesos-containerizer

|—mesos-fetcher

|—mesos-health-check

|—mesos-usage

|—mesos-logrotate-logger

## 3.3启动mesos进程

**注：在root用户下执行**

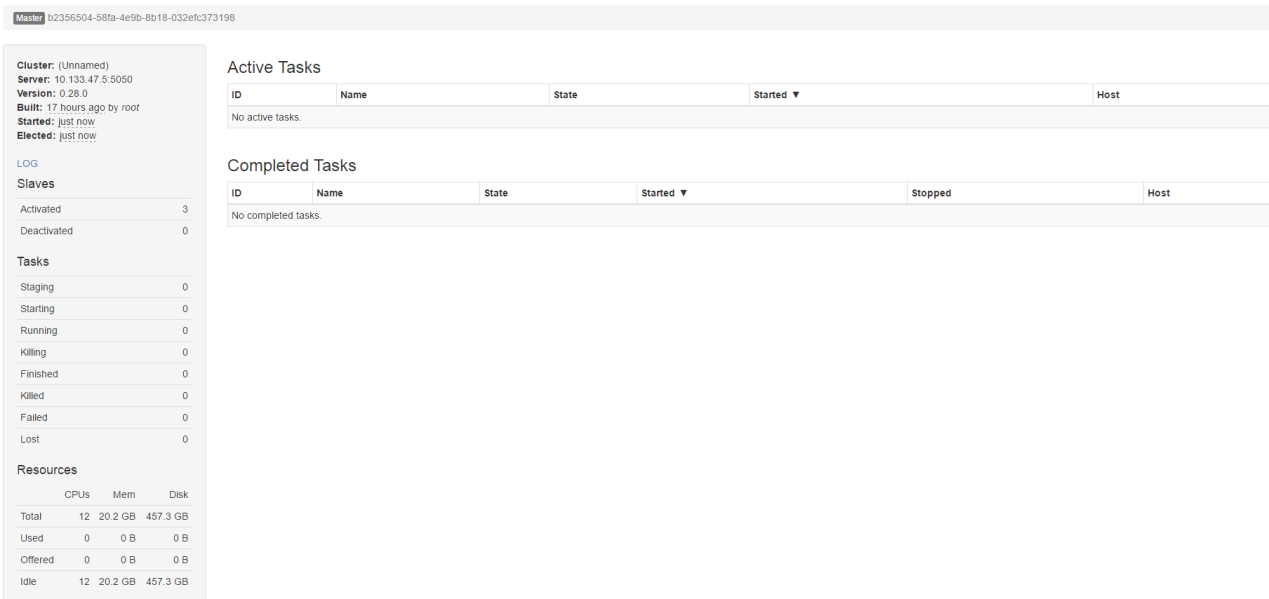
# 启动master进程，master上执行命令

$/usr/local/sbin/mesos-start-masters.sh

# 在各主机上启动slave进程, 每台slave上执行命令

$ nohup mesos-slave --master=fys1.cmss.com:5050 --log\_dir=/home/mesos/log/mesos/slave/log --work\_dir=/home/mesos/data/mesos/slave/data &

启动完毕后，查看master主机的5050端口页面



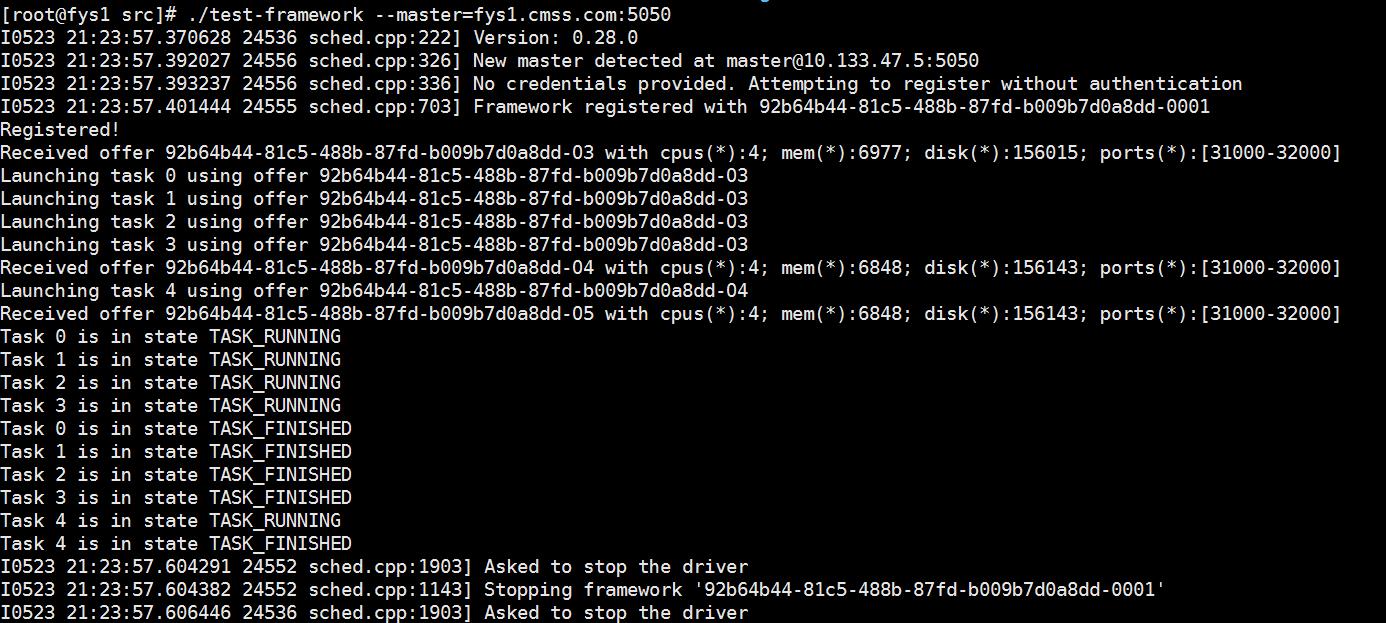
## 3.4 Mesos的使用

### 3.4.1 Framework example

1）执行 C++ framework example

./src/test-framework --master=127.0.0.1:5050

结果如下：



页面上显示结果如下：



2）运行Java framework

$ ./src/examples/java/test-framework fys1.cmss.com:5050

页面上显示结果：



3）运行python framework

$ ./src/examples/python/test-framework fys1.cmss.com:5050

### 3.4.2 Framework On Mesos

1. Hadoop On Mesos

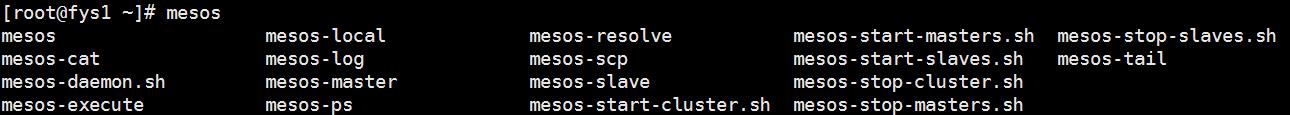
2.Spark On Mesos

3.Strom On Mesos

https://github.com/mesos/storm

### 3.4.3 命令行

mesos脚本如下所示



# 四、性能测试

# 五、源码分析

# 总结