

Docker on Yarn 现状

大数据部



我们面临的问题

我们
面临的问题



任务之间的环境隔离



任务之间的资源隔离



节点的CPU资源控制



任务环境的快速变更

我们的方案选择

1

主机上安装
不同版本的
基础环境

主机环境混乱

2

Docker运
行在Yarn框
架上实现环
境隔离

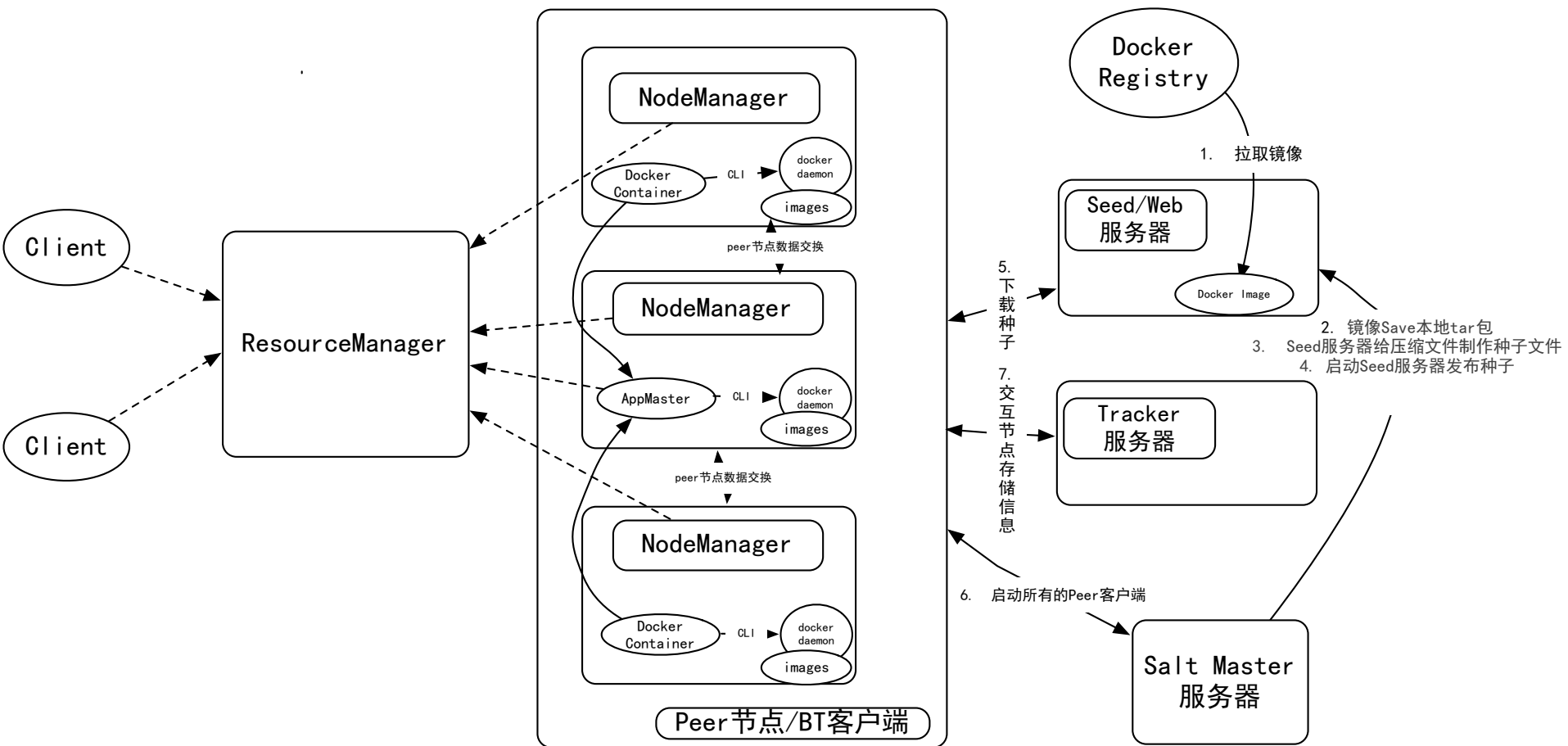
Docker on Yarn

3

任务自行分
发基础环境
的包

GCC包的问题

总体架构图



我们需要解决的问题

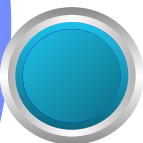
我们需要
解决的问题



如何构建一个Docker镜像



如何存储一个Docker镜像



如何分发一个Docker镜像



DockerContainer的资源隔离和环境隔离


Docker镜像的构建




Docker镜像构建上线情况


Jenkins自动构建镜像


实例: hecate集群 mart_dm_tbi 用户


 **Jenkins**


Jenkins ▶ Hecate_Tbi_Dockerfile ▶ #44


 返回到工程


 状态集


 变更记录


 **Console Output**

 View as plain text

 编辑编译信息

 删除本次生成

 前一次构建

 控制台输出

跳过 247 KB.. [完整日志](#)

ng build context to Docker daemon 2.781 GB
Sending build context to Docker daemon 2.782 GB
Sending build context to Docker daemon 2.782 GB
Sending build context to Docker daemon 2.783 GB
Sending build context to Docker daemon 2.784 GB
Sending build context to Docker daemon 2.784 GB
Sending build context to Docker daemon 2.785 GB
Sending build context to Docker daemon 2.785 GB
Sending build context to Docker daemon 2.786 GB
Sending build context to Docker daemon 2.786 GB
Sending build context to Docker daemon 2.787 GB
Sending build context to Docker daemon 2.788 GB

Docker镜像构建上线情况

Docker镜像环境的验证

实例: hecate集群 mart_dm_tbi 用户

bash

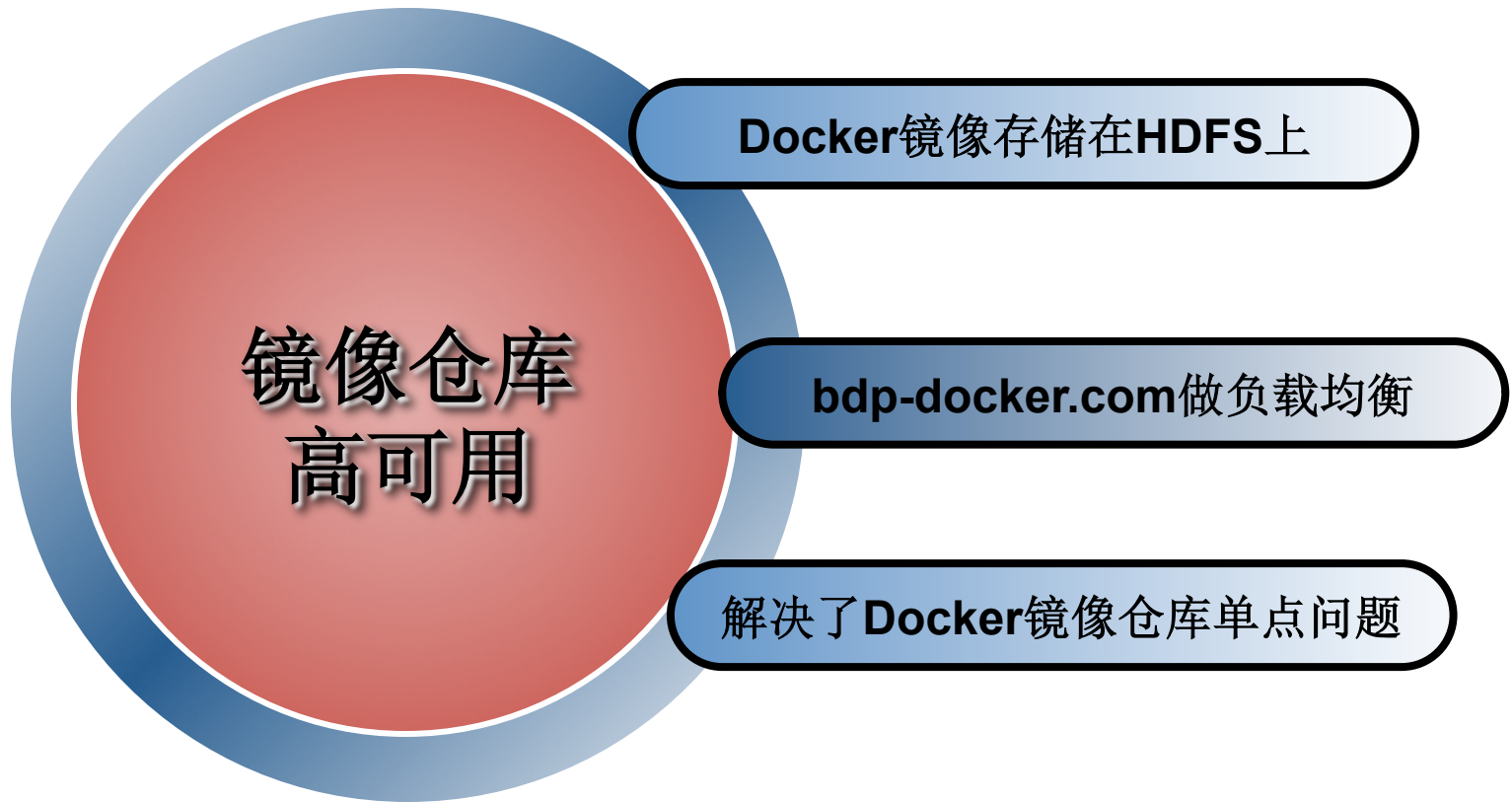
>_

连容器终端

退出容器终端

```
[root@0977a1fff104 /]# python
Python 2.7.6 (default, Mar 22 2016, 22:12:34)
[GCC 4.8.3 20140911 (Red Hat 4.8.3-9)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>>
```


Docker镜像的存储



Docker镜像的分发

镜像 分发难点




镜像大、要分发的节点多



用普通的HTTP 和FTP下载耗时很长



镜像文件的完整性的保障



单个节点的带宽占满

Docker镜像的分发

BT部署

BitTorrent协议(简称BT)是一种对等网络中文件共享的网络协议。本项目依赖于BT协议，在twitter开源代码murder基础上进行修改，在集群中实现了文件的快速分发。

与传统的HTTP/FTP下载相比，BT不会因为下载人数过多导致服务器带宽满负荷。本项目在Twitter项目基础上进行开发，实验表明，在353台部署4.1G大小的文件，仅耗时4分35秒。如果以100M/s进行计算，使用传统方法为353台机器部署4.1G的文件的时间将达到247分钟，提速54倍。



Docker镜像分发



BT部署 的优势

1 快速

BT的引入大大减小了部署的时间。部署时间不会随集群规模的增大而增大。

2 安全

Peer会计算下载的块的hash值是否与种子中对应的hash值相同。保证文件完整性

3 可配置

本项目能够对各Peer节点进行控制，实现对网络带宽、上传连接数目等参数的限制。

4 多TRACKER

当peer节点过多，出现tracker服务器压力过大的时候，本项目可以通过配置多个tracker缓解节点压力。

DockerContainer的运行

如何运行
DockerContainer



如何切换DockerContainer



如何隔离DockerContainer的CPU



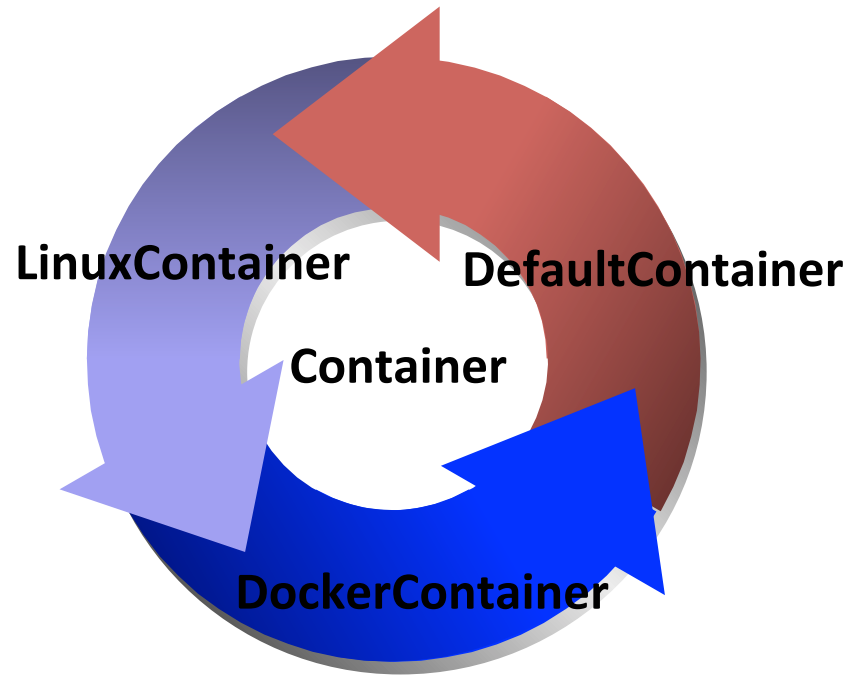
如何隔离DockerContainer的Memory



DockerContainer上线情况

DockerContainer的运行

客户端Container 的无缝切换

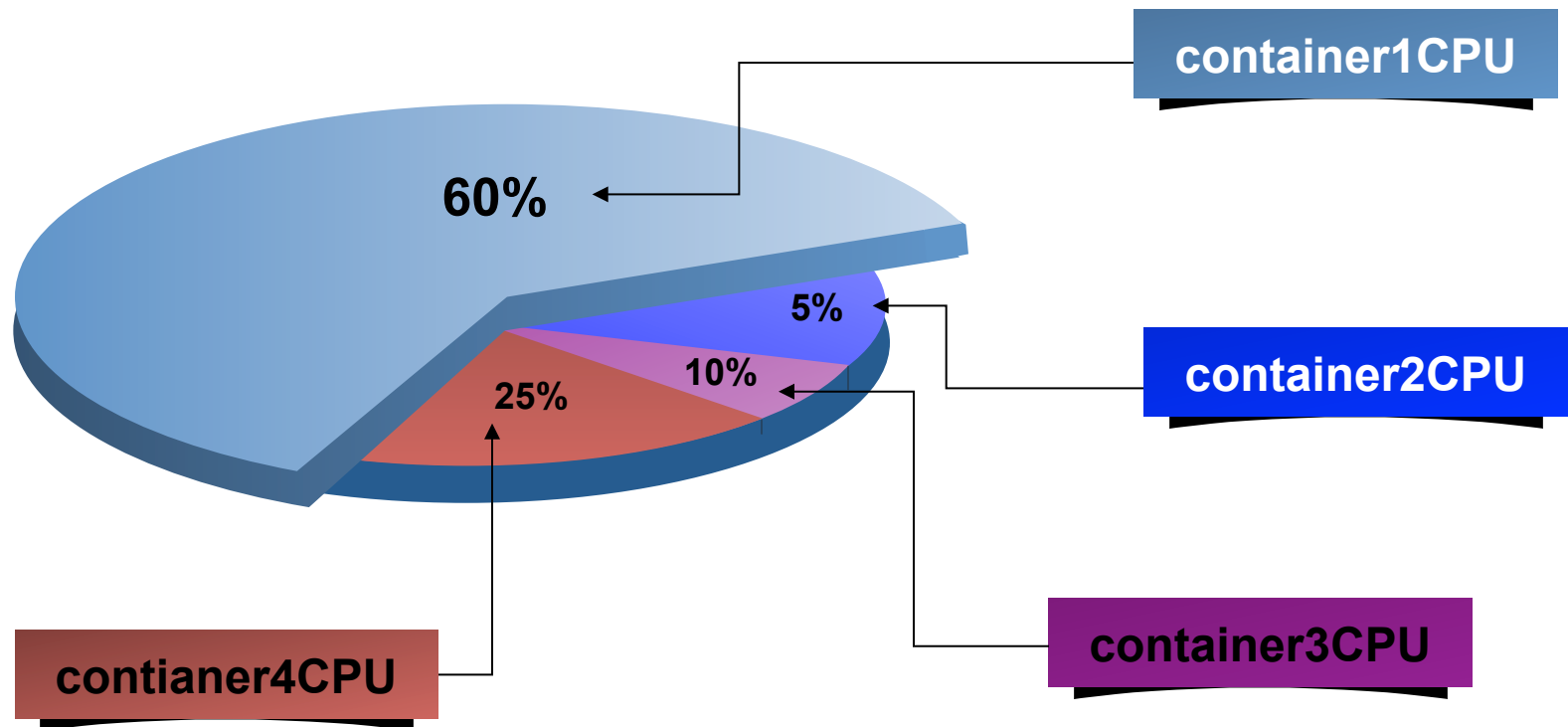


- 1、改造ContainersLaunch实现了三种不同类型Container在客户端无缝切换。
- 2、增加三种Container实例的缓存提高了不同Container的Launch。

DockerContainer的运行

DockerContainer 的CPU隔离

$\text{ContainerCPU} = \text{ContainerVCores} * \text{nodeCpuPercentage} * \text{numProcessor} / \text{NodeVCores}$



修复了 [Yarn-2194 Hadoop CentOS7利用Cgroup的Bug]
(<https://issues.apache.org/jira/browse/YARN-2194>)

DockerContainer的运行

DockerContainer 的Memory隔离



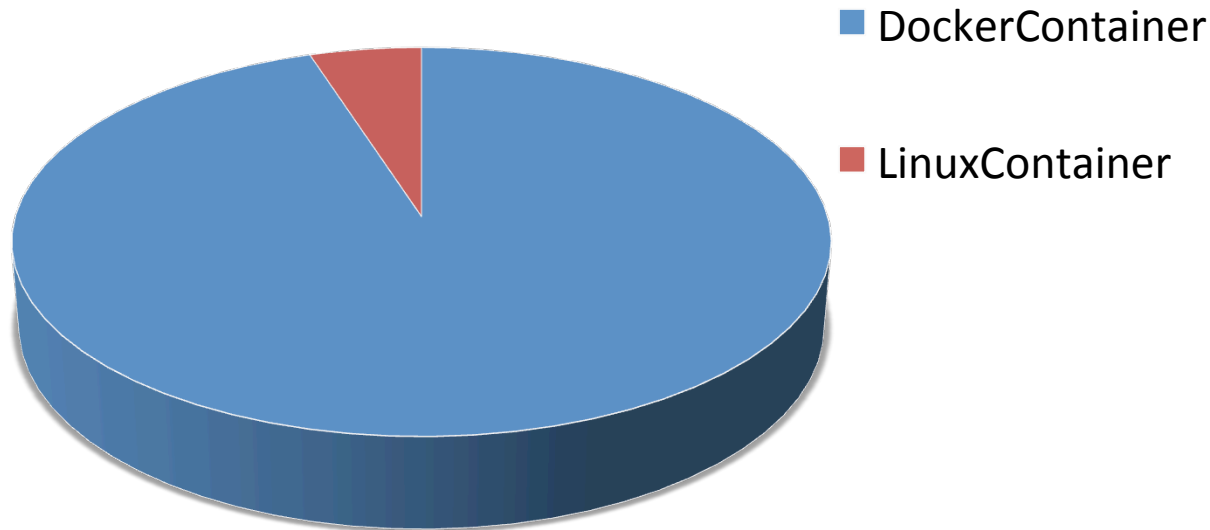
- 1、调研了 [YARN-1856 Cgroups内存隔离] (<https://issues.apache.org/jira/browse/YARN-1856>)
- 2、修复了 [YARN-3080 DockerContainer获取不了Pid的Bug] (<https://issues.apache.org/jira/browse/YARN-3080>)

DockerContainer上线情况

截至目前为止

Hecate集群任务95%已经运行在DockerContainer上。

Driud集群已经全部运行LinuxContainer上，利用Cgroup控制CPU资源。



DockerContainer上线情况

MapReduce 运行在DockerContainer上



Configuration for MapReduce Job job_1467889878374_2036


Logged in as: dr.who

hdfs://ns1:8020/user/history/done/2016/07/15/000002/job_1467889878374_2036_conf.xml

Show 20 entries		Search: docker
key	value	source chain
mapreduce.map.env	yarn.nodemanager.docker-container-executor.image-name=bdp-docker.jd.com:5000/hecate_base_dmp:0.0.7,yarn.nodemanager.container-executor.class=org.apache.hadoop.yarn.server.nodemanager.DockerContainerExecutor	job.xml ← mapred-site.xml
mapreduce.reduce.env	yarn.nodemanager.docker-container-executor.image-name=bdp-docker.jd.com:5000/hecate_base_dmp:0.0.7,yarn.nodemanager.container-executor.class=org.apache.hadoop.yarn.server.nodemanager.DockerContainerExecutor	job.xml ← mapred-site.xml
yarn.app.mapreduce.am.env	yarn.nodemanager.docker-container-executor.image-name=bdp-docker.jd.com:5000/hecate_base_dmp:0.0.7,yarn.nodemanager.container-executor.class=org.apache.hadoop.yarn.server.nodemanager.DockerContainerExecutor	job.xml ← mapred-site.xml
yarn.nodemanager.docker-container-executor.exec-name	/usr/bin/docker	job.xml ← yarn-default.xml
key	value	source chain
Showing 1 to 4 of 4 entries (filtered from 799 total entries)		First Previous 1 Next Last

DockerContainer上线情况

Spark 运行在DockerContainer上

 1.5.2

JobsStagesStorageEnvironmentExecutors

HKMeans_d6.15.local (application... application UI)

Environment

Runtime Information

Name	Value
Java Home	/software/servers/jdk1.7.0_67/jre
Java Version	1.7.0_67 (Oracle Corporation)
Scala Version	version 2.10.4

Spark Properties

Name	Value
spark.shuffle.maxMergeFactor	50
spark.executorEnv.yarn.nodemanager.docker-container-executor.image-name	bdp-docker.jd.com:5000/hecate_base_dmp:0.0.7
spark.serializer	org.apache.spark.serializer.KryoSerializer
spark.kryoserializer.buffer.max.mb	512
spark.speculation	true
spark.executor.extraJavaOptions	-verbose:gc -Xss512m -XX:+UseCompressedOops -XX:-PrintGCDetails -XX:+PrintGCTimeStamps -XX:CMSInitiatingOccupancyFraction=60 -Djava.library.path=/data0/cv/lib

DockerContainer上线情况

Docker Container CPU极限测试: (CPU-Limit 为60)

自定义程序说明:

com.jd.bdp.hadoop.mapreduce.CPUTest

该程序的Map生成无限空循环的线程

```
top - 10:55:10 up 65 days, 19:23, 1 user, load average: 308.02, 750.93, 399.20
Tasks: 461 total, 1 running, 458 sleeping, 0 stopped, 2 zombie
%Cpu0  : 60.9 us,  0.0 sy,  0.0 ni, 39.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  : 60.1 us,  0.0 sy,  0.0 ni, 39.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2  : 60.4 us,  0.0 sy,  0.0 ni, 39.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3  : 59.4 us,  0.0 sy,  0.0 ni, 40.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu4  : 61.4 us,  0.0 sy,  0.0 ni, 38.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu5  : 60.1 us,  0.0 sy,  0.0 ni, 39.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu6  : 59.9 us,  0.0 sy,  0.0 ni, 40.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu7  : 60.9 us,  0.0 sy,  0.0 ni, 39.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu8  : 59.5 us,  0.0 sy,  0.0 ni, 40.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu9  : 60.1 us,  0.0 sy,  0.0 ni, 39.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu10 : 60.4 us,  0.0 sy,  0.0 ni, 39.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu11 : 61.1 us,  0.0 sy,  0.0 ni, 38.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu12 : 60.5 us,  0.0 sy,  0.0 ni, 39.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu13 : 61.2 us,  0.3 sy,  0.0 ni, 38.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu14 : 60.5 us,  0.0 sy,  0.0 ni, 39.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu15 : 59.7 us,  0.0 sy,  0.0 ni, 40.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu16 : 60.9 us,  0.0 sy,  0.0 ni, 39.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu17 : 60.9 us,  0.0 sy,  0.0 ni, 39.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu18 : 61.2 us,  0.0 sy,  0.0 ni, 38.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu19 : 60.2 us,  0.0 sy,  0.0 ni, 39.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu20 : 61.6 us,  0.3 sy,  0.0 ni, 38.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu21 : 60.2 us,  0.0 sy,  0.0 ni, 39.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu22 : 60.9 us,  0.0 sy,  0.0 ni, 39.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu23 : 60.9 us,  0.0 sy,  0.0 ni, 39.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu24 : 61.1 us,  0.0 sy,  0.0 ni, 38.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu25 : 61.1 us,  0.3 sy,  0.0 ni, 38.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu26 : 60.9 us,  0.0 sy,  0.0 ni, 39.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu27 : 60.7 us,  0.0 sy,  0.0 ni, 39.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu28 : 60.5 us,  0.0 sy,  0.0 ni, 39.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu29 : 60.9 us,  0.0 sy,  0.0 ni, 39.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu30 : 60.1 us,  0.0 sy,  0.0 ni, 39.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu31 : 60.4 us,  0.0 sy,  0.0 ni, 39.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 26385889+total, 7530024 free, 5361664 used, 25096720+buff/cache
KiB Swap: 16777212 total, 16777212 free, 0 used. 25751153+avail Mem

  PID USER      PR  NI   VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
23638 yarn      20   0 4781476 846356 17292 S   19.2    0.3 85:12.88 java
30372 yarn      20   0 1841552 338232 17708 S    1.0    0.1 14:49.92 java
26265 root        20   0 146380 2436 1440 R    0.3    0.0 0:00.04 top
```

Docker on Yarn 后续计划

后续计划



Yarn 支持更多的资源调度



整个Docker on Yarn流程优化



Docker镜像用户自助式服务

Docker on Yarn 后续计划

序号	任务	优先级	用时	进度	起始日期	参与者
1	Yarn的网络IO调度的支持					
	1、NodeManager 支持Cgroup 的网络资源隔离					
	2、在container-executor增加tc的支持					
	3、修改Yarn资源调度模型					
	4、在FairScheduler 上增加网络资源调度					
	5、在MRAppMaster 上增加网络资源调度					
	6、在YarnClient 上增加网络资源调度					
	7、在WebUI & Metrics 上增加网络资源调度					

谢谢！

