大型架构及配置技术

Ansible

一款自动化运维工具，基于python研发，可实现批量操作系统配置，批量部署，批量运行命令。

调用特定的模块完成特定的任务

部署简单，主从模式工作，支持自定义模块，支持playbook，易于使用，支持多层部署，支持异构it环境。

Ansible作用及工作结构

ansible是基于模块工作的，本身没有批量部署的能力，真正具有批量部署的是ansible所允许的模块，ansible提供框架。

主要包括：

1）连接插件connection plugins：负责和被监控端实现通信；

2）host inventory：指定操作的主机，是一个配置文件里面定义监控的主机；

3）各种模块核心模块，command模块，自定义模块；

4）借助于插件完成记录日志邮件等功能；

5）playbook：剧本执行多个任务时，非必须可以让节点一次性运行多个任务。

ansible执行时两个重点：主机集合和命令集合

Ansible安装

软件依赖python，需要使用paramiko，pyYAML，jinja2，httplib2，six模块，所以都需要安装。

ansible默认通过ssh协议管理机器，被管理主机需要开启ssh服务，且允许ansible主机连接。

ansible --version

环境准备

启动6台虚拟机（base-vm创建）

ansible 192.168.1.40 管理主机

web1 192.168.1.41 托管主机

web2 192.168.1.42 托管主机

db1 192.168.1.43 托管主机

db2 192.168.1.44 托管主机

cache 192.168.1.45 托管主机

主机定义与分组

ansible配置文件/etc/ansible/ansible.cfg

14:inventory定义托管主机地址配置文件路径

61:host\_key\_checking去掉注释，防止远程时输入yes

主机分组定义

修改主机名映射/etc/hosts

vim /etc/ansible/hosts

末行添加

[组名称]

主机名称或IP地址，其它参数

[web]

web1

web2

[db]

db[1:10] db1到db10

Ansible命令基础

ansible 主机集合 -m 模块名称 -a 模块参数

主机集合：主机名或分组名，多个使用逗号分隔

-m：模块名称，默认command模块

-a：模块参数

-i：inventory文件路径，或可执行脚本

-k：使用交互式登录密码

-e：定义变量

-v：显示详细信息

ansible web，db --list-hosts 查看web和db组成员

ansible all --list-hosts 列出所有主机

部署证书文件

ansible是通过ssh远程执行命令

ssh-keygen -t rsa -f "/root/.ssh/key" -b 2048 -N ''

ssh-copy-id -i /root/.ssh/key.pub 主机

ssh -i key 主机

inventory扩展参数

ansible\_ssh\_port

ssh端口号：如果不是默认的端口号，通过变量设置

ansible\_ssh\_user

默认的ssh用户名

ansible\_ssh\_pass

ssh密码（这种方式并不安全，建议使用--ask-pass或ssh密钥）

ansible\_ssh\_private\_key\_file

ssh使用的私钥文件，适用于有多个密钥，而你不想使用ssh代理的情况

分组定义

[other]

cache

[all:vars] 对组other定义变量

ansible\_ssh\_private\_key\_file="/root/.ssh/key"

对单独主机声明变量主机后面空格添加

子组定义

[app:children]

web

db

自定义分组文件

mkdir /var/kk

cd /var/kk

vim ansible.cfg

[defaults]

inventory = myhosts

host\_key\_checking = False

vim myhosts

[app1]

web1

web2

:wq

验证 ansible app1 --list-hosts

ansible-doc 模块帮助手册

ansible -l 列出所有模块

command模块

ansible web -m command -a 'uptime'

shell模块

shell模块是通过/bin/bash执行命令，可以执行任意命令，但是不能执行交互式命令。

练习题：给web1 db2添加用户nb，设置nb密码为123

ansible web1,db2 -m shell -a 'useradd nb && echo 123 | passwd --stdin nb'

思考题：给所有主机添加用户wk，要求nb用户与wk用户不能出现再同一台主机上，设置wk用户的密码是456

ansible all -m shell -a 'id nb || useradd wk && echo 456 | passwd --stdin wk'

script模块

在本地写脚本，使用ansible批量执行

vim user.sh

#!/bin/bash

id nb

if [ $? != 0 ];then

useradd wk

echo 456 | passwd --stdin wk

fi

ansible all -m script -a './user.sh'

yum模块

使用yum包管理器来管理软件包

name：要进行操作的软件包的名字

state：动作（installed ， removed）

例：ansible db -m yum -a 'name=mariadb-server state=installed' 给db组主机安装mariadb

service模块

name：必选项，服务名称

enabled：是否开机启动yes|no

sleep：执行restarted，会在stop和start之间沉睡几秒钟

state：对当前服务执行启动，停止，重启，重新加载等操作（started，stoped，restarted，reloaded）

ansible cache -m service -a ‘name=sshd enabled=yes state=started’

ansible db -m service -a 'name=mariadb state=started enabled=yes'

copy模块

复制文件到远程主机

src：复制本地文件到远程主机，绝对路径和相对路径都可以，路径为目录时会递归复制。

若路径以”/“结尾，只复制目录里的内容，若不以”/“结尾，则复制包含目录在内的整个内容，类似于rsync。

dest：必选项，远程主机的绝对路径，如果源文件是一个目录，那该路径必须是目录。

backup：覆盖前先备份原文件，备份文件包含时间信息，有两个选项：yes|no

force：若目标主机包含该文件，但内容不同，如果设置为yes，则强制覆盖；

设为no，则只有当前目标主机的目标位置不存在该文件时才复制，默认为yes。

复制文件：ansible all -m copy -a ‘src=/etc/resolv.conf dest=/etc/resolv.conf’

复制目录：ansible all -m copy -a 'src=/etc/yum.repos.d/ dest=/etc/yum.repos.d/ '

练习题：给所有db主机开启binlog日志(scp db1:/etc/my.cnf .)

log\_bin = mysql-bin

binlog-format = mixed

ansible all -m copy -a 'src=./my.cnf dest=/etc/my.cnf'

lineinfile模块

类似sed的一种行编辑替换模块

path 目标文件文件

regexp 正则表达式，要修改的行

line 最终修改的结果

例：my.cnf 中 bin-log的格式

ansible db -m lineinfile -a ' path="/etc/my.cnf" regexp="^binlog-format" line="binlog-format = row"'

replace模块

只修改匹配到的

ansible db -m lineinfile -a 'path="/etc/my.cnf" regexp="row$" replace="mixed"'

setup模块

setup模块主要用于获取主机信息，playbooks里经常会用的另一个参数gather\_facts与该模块相关，setup模块下经常用的filter参数

filter过滤所需信息

ansible cache -m setup -a 'filter=adsible\_distribution'

ansible七种武器

第一种武器

ansible命令，用于执行临时性的工作，必须掌握

第二种武器

ansible-doc时ansible模块的文档说明，针对每个模块都有详细的说明及应用案例介绍。

第三种武器

ansible-console是ansible为用户提供的交互式工具，用户可以再ansible-console虚拟出来的终端上像shell一样使用ansible内置的各种命令

这为习惯使用shell交互式的用户提供了良好的使用体验。

第四种武器

ansible-galaxy从github上下载管理Roles的一款工具，与python的pip类似

第五种武器

ansible-playbook是日常应用中使用频率最高的命令，工作机制：通过读取编写好的playbook文件实现批量管理

可以理解为按一定条件组成的ansible任务集，必须掌握。

第六种武器

ansible-vault主要用于配置文件加密，如编写的playbook文件中包含敏感信息，不想其他人随意查看，可用它加密/接密这个文件

第七种武器

ansible-pull

ansible有两种工作模式pull/push，默认使用push模式工作，pull和push工作模式机制刚好相反

适用场景：有大批量机器需要配置，即便使用高并发线程依旧要花费很多时间

通常在配置大批量机器的场景下使用，灵活性稍有欠缺，但效率几乎可以无限提升，对运维人员的技术水平和前瞻性规划有较高的要求。

json简介

json是javascript对象表示法，它是一种基于文本独立于语言的轻量级数据交换格式

json中的分隔符限于单引号”'“，小括号”（）“，中括号”[]“,大括号”{}“,冒号”:“和逗号”，“

json特性

json是纯文本

json具有”自我描述性“

json具有层级结构

json可通过javascript进行解析

json语法规则

数据在名称/值对中

数据由逗号分隔

大括号保存对象

中括号保存数组

json数据的书写格式是：名称/值对

名称/值对包括字段名称（在双引号中），后面写一个冒号，然后是值，例如：

“诗仙”：“李白”

json语法规则之数组

{ "诗人":

["李白","杜甫","白居易","李贺"]

}

复合复杂类型

{ "诗人":

[{"李白":"诗仙","年代":"唐"},

{"杜甫":"诗圣","年代":"唐"},

{"白居易":"诗魔","年代":"唐"},

{"李贺":"诗鬼","年代":"唐"}

]

}

yaml简介

是一个可读性高，用来表达数据序列的格式

yaml参考了多种语言，如：C语言，python，perl等，并从xml，电子邮件的数据格式中获得灵感，clark Evans在2001年首次发表了这种语言，目前已有数种编程语言或脚本语言支持这种语言。

YAML基础语法

YAML的结构通过空格来展示

数组使用“-”来表示

键值对使用“：”来表示

YAML使用一个固定的缩进风格表示数据层级结构关系

一般每个缩进级别由两个以上空格组成

#表示注释

注意：

不要使用tab，缩进是初学者容易出错的地方之一

同一层级缩进必须对齐

yaml的键值表示方法

采用冒号分隔

：后面必须有一个空格

yaml键值对例子

"诗仙":"李白"

-或

"李白":

"诗仙"

复杂YAML的键值对嵌套

"诗人":

"李白":"诗仙"

或

"诗人":

"李白":

"诗仙"

数组

["李白","杜甫","白居易","李贺"]

YAML数组表示方法

使用一个段横杠加一个空格

YAML数组例子

- "李白"

- "杜甫"

- "白居易"

- "李贺"

哈希数组符合表达式

"诗人":

- "李白"

- "杜甫"

- "白居易"

- "李贺"

高级复合表达式

"诗人":

-

"李白":"诗仙"

"年代":"唐"

-

"杜甫":"诗圣"

"年代":"唐"

-

"白居易":"诗魔"

"年代":"唐"

-

"李贺":"诗鬼"

"年代":"唐"

jinja2模板简介

jinja2是基于python的模板引擎，包含变量和表达式两部分，两者在模板求值时会被替换为值，模板中还有标签，控制模板的逻辑

playbook的模板使用python的jinja2模块来处理

jinja2模板的表达式都是包含在分隔符"{{}}"内的

控制语句都是包含在分隔符"{% %}"内的

模板支持注释，都是包含在分隔符"{# #}"内，支持块注释

调用变量

{{varname}}

计算

{{2+3}}

判断

{{1 in [1,2,3]}}

jinja2模板控制语句

{% if name == '诗仙' %}

李白

{% elif name == '诗圣' %}

杜甫

{% elif name == '诗魔' %}

白居易

{% else %}

李贺

{% endif %}

jinja2模板控制语句

{% if name == '诗仙' %}

李白

{% elif name == '诗圣' %}

杜甫

{% elif name == '诗魔' %}

白居易

{% elif name == '于谦' %}

{% for method in [抽烟,喝酒,烫头]%}

{{do method}}

{% endfor %}

{% enif %}

jinja2过滤器

变量可以通过过滤器修改，过滤器与变量用管道符号(|)分割，也可以用圆括号传递可选参数，多个过滤器可以链式调用

前一个过滤器的输出会被作为后一个过滤器的输入

例如

加密一个字符串:"{{ 'astr'|password\_hash('sha512')}}"

playbook

playbook时ansible用于配置，部署和管理托管主机的剧本，通过playbook的详细描述，执行其中的一系列tasks

可以让远端的主机达到预期状态；

也可以说，playbook字面意思即剧本，现实中由演员按剧本表演，在ansible中由计算机进行安装，部署应用，提供对外服务

以及组织计算机处理各种各样的事情

为什么要使用playbook

执行一些简单的任务，使用ad-hoc命令可以方便的解决问题，但有时一个设施过于复杂时，执行ad-hoc命令是不合适的，最好使用playbook

playbook可以反复使用编写的代码，可以放到不同的机器上面，像函数一样，最大化的利用代码，在使用ansible的过程中，处理大部分操作都是在编写playbook

playbook语法基础

playbook由YAML语言编写，遵循YAML标准

在同一行中，#之后的内容表示注释

同一个列表中的元素应该保持相同的缩进

playbook由一个或多个play组成

play中hosts，variables，roles，tasks等对象的表示方法是键值中间以":"分隔表示

YAML还有一个小得怪癖，它得文件开始行都应该是 ---，这是YAML格式得一部分，表明一个文件得开始

playbook构成

hosts：定义将要执行playbook得远程主机组

vars：定义playbook运行时需要使用得变量

tasks：定义将要在远程主机上执行得任务列表

handlers：定义task执行完成以后需要调用得任务

playbook执行结果

使用ansible-playbook运行playbook文件，输出内容为json格式，由不同颜色组成便于识别

-绿色表示执行成功

-\*\*\*代表系统状态发生改变

-红色代表执行失败

第一个playbook

vim myping.yml

--- #第一行，表示开始

- hosts: web

- hosts: db

remote\_user: root

tasks:

- ping:

ansible-playbook myping.yml -f 5

rm -rf .ansible/cp/\* 清除ansible缓存

-f 并发进程数量，默认是5

hosts行，内容是一个(多个)组或主机得patterns，以逗号为分隔符

remote\_user 账户名

tasks

命令得集合

每一个play包含了一个tasks列表（任务列表）

一个task在其所对应得所有主机上（通过host pattern 匹配得所有主机）执行完毕之后，下一个task才会执行

hosts

主机的集合

定义要执行任务得主机

playbook执行命令

给web主机添加用户z3

设置默认密码123，并z3用户添加到users组

---

- hosts: web

remote\_user: root

tasks:

- name: create user z3

user:

name: z3

group: users

- name: set password

shell: echo 123 | passwd --stdin z3

vim可视块操作：Ctrl+v进入可视块模式，通过上下左右键选择块，shift+i 进入操作，Esc完成

playbook练习

1，所有web主机安装apache

2，修改配置文件得监听端口为8080

3，设置默认主页hello world

4，启动服务

5，设置开机自启

---

- hosts: web

remote\_user: root

tasks:

- name: install the latest version of Apache

yum:

name: httpd

state: latest

- lineinfile:

path: /etc/httpd/conf/httpd.conf

regexp: '^Listen '

insertafter: '^#Listen '

line: 'Listen 8080'

- copy:

src: index.html

dest: /var/www/html/index.html

owner: apache

group: apache

mode: 0644

- service:

name: httpd

state: started

enabled: yes

playbook进阶

变量

添加用户

给web主机添加用户l4，设置密码123

---

- hosts: web

remoute\_user: root

vars:

username: l4

tasks:

- name: create user "{{username}}"

user:

name:"{{username}}"

- shell: echo 123 | passwd --stdin "{{username}}"

设密码

解决密码明文问题

user模块得password为什么不能设置密码呢

经过测试发现，password是把字符串直接写入

shadow，并没有改变，二Linux得shadow密码是经过加密得，所以不能使用

解决方案

变量过滤器password\_hash

{{'urpassword' | password\_hash('sha512')}}

变量过滤器使用

给db主机添加用户plj，设置默认密码123qqq...A

- hosts: db

remote\_user: root

vars:

username: plj

tasks:

name: create user "{{username}}"

user:

password={{'123qqq...A' | password\_hash('sha512')}}

name={{username}}

变量参数

传递参数

-e 参数

参数格式必须是json 或 yaml

yaml 格式 可以使用参数文件，例如

vim args.yml

---

username:

nb

ansible-playbook user.yml -e @args.yml

或

ansible-playbook user.yml -e {"username":"plj"}

变量练习

1.练习使用user模块添加用户

2.练习使用变量简化task，让play通用性更强

3.练习使用过滤器

error

ansible-playbook对错误得处理

默认情况判断$?,如果值不为0就停止执行

但某些情况我们需要忽略错误继续执行

例如 创建缓存目录，然后重启apache

---

- hosts: cache

remote\_user: root

tasks:

-shell: mkdir /tmp/cache

- name:ReStart service httpd

service:

name: httpd

state: restarted

错误处理方法

ignore\_errors: 对错误得处理方式

Trun 表示忽略错误继续执行

False 表示遇到错误就停止执行

默认 False

处理方式

-name: run some command

shell: /usr/bin/somecommand

ignore\_errors: True

tags

tags:给指定得任务定义一个调用表示

使用格式：

- name: NAME

- module: arguments

- tags: TAG\_ID

playbook 调用方式

-t TAGS， --tags=TAGS

修改httpd配置得 yml

tasks:

- copy:

src: httpd.conf

dest: /etc/httpd/conf/httpd.conf

owner: apache

group: apache

mode: 0644

- copy

src: index.html

dest: /var/www/html/index.html

owner: apache

group: apache

mode: 0644

设置tags标记

tasks:

-copy:

src: httpd.conf

dest: /etc/httpd/conf/httpd.conf

owner: apache

group: apache

mode: 0644

tags: syncconf

调用标记

ansible-playbook conf.yml -t syncconf

handlers

当关注得资源发生变化时采取得操作

notify这个action可用于在每个play得最后被出发，这样可以避免有多次改变发生时每次都执行指定得操作

取而代之仅在所有得变化发生完成后一次性地执行指定操作

在notify中列出得操作称为handler，即notify调用handler中定义得操作

前面安装了apache，修改httpd得配置文件，重新载入配置文件让服务生效。

handlers:

- name: restart apache

service:

state: restarted

结合之前实验，使用handlers

- copy

src: httpd.conf

dest: /etc/httpd/conf/httpd.conf

owner: apache

group: apache

mode: 0644

tags: syncconf

notify:

- restart httpd

handlers:

- name: restart httpd

service:

name: httpd

state: restarted

handlers注意事项

notify调用得是handler段得name定义得串，必须一致，否则达不到出发得效果

多个task触发同一个notify得时候，同一个服务只会触发一次

notify可以出发多个条件，在生产环境中往往涉及到某一个配置文件得改变要重启若干服务得场景，handler用到这里非常合适

结合vars可以写出非常普适得服务管理脚本

休息带练习到21：15

when

满足特定条件触发操作

register

变量注册

针对运行命令结果得返回值做判断

vim load.yml

---

- hosts: web

remote\_user: root

tasks:

- shell: uptime | awk '{printf(''%.2f\n",$(NF-2))}'

register: result

- service:

name: httpd

state: stopped

when: result.stdout | float > 0.7

通过awk 'BEGIN{while(1){}}' & 提高系统负载，验证实验

with\_items

是playbook标准循环，可以用于迭代一个列表或字典，通过{{item}}获取每次迭代得值

例如创建多个用户（有使用tab建，不能复制）

---

- hosts: db

remote\_user: root

tasks:

- name: add users

user: {{item.username}}

group: {{item.group}}

password: {{item.passwd| password\_hash('sha512')}}

with\_items:

-

username : nb

group: users

passwd: "123456"

-

username: wk

group: bin

passwd: banana

-

username: dd

group: apache

passwd: "123"

-

username: plj

group: mysql

passwd: 123qqq...A

-

username: xx

group: ssh

passwd: a

ELK

什么是ELK

ELK是一整套解决方案，是三个软件产品得首字母缩写，很多公司都在使用，如：Sina，携程，华为，美团等

ELK分别代表

-Elasticsearch：负责日志检索和储存

-Logstash：负责日志得收集和分析，处理

-Kibana：负责日志得可视化

ELK能做什么

ELK组件在海量日志系统得运维中，可用于解决

-分布式日志数据集中式查询和管理

-系统监控，包含系统硬件和应用各个组件得监控

-故障排查

-安全信息和事件管理

-报表功能

Elasticsearch

是一个基于Lucene得搜索服务器，它提供了一个分布式多用户能力得全文搜索引擎，基于RESTful API得web接口

Elasticsearch是用于java开发得，并作为Apache许可条款下得开放源码发布，是当前流行得企业级搜索引擎，设计用于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。

主要特点

-实时分析

-分布式实时文件存储，并将每一个字段都编入索引

-文档导向，所有得对象全部是文档

-高可用性，易扩展，支持集群（cluster），分片和复制（shards和replicas）

-接口友好，支持json

Elasticsearch没有什么

-ES没有典型意义得事务

-ES是一种面向文档得数据库

-ES没有提供授权和认证特性

Elasticsearch

-Node：装有一个ES服务器得节点

-Cluster：有多个Node组成得集群

-Document：一个可被搜索得基础信息单元

-Index：拥有相似特征得文档得集合

-Type：一个索引中可以定于一种或多种类型

-Filed：是ES得最小单位，相当于数据得某一列

-Shards：索引得分片，每一个分片就是一个shard

-Replicas：索引的拷贝

SQL与NoSQL

ES与关系型数据库得对比

-在ES中，文档属于一种类型（Type），而这些类型存在于索引（index）中，类比传统关系型数据库

- DB --> Databases --> Tables --> Rows --> Columns

- 关系型 数据库 表 行 列

- ES --> Indices --> Types --> Documents --> Fileds

- ES 索引 类型 文档 域（字段）

ES集群安装

安装第一台ES服务器

-设置主机名称和IP对应关系

配置/etc/hosts

192.168.1.51 es1

-解决依赖关系

Elasticsearch要求至少java7

一般推荐使用OpenJDK 1.8

yum -y install java-1.8.0-openjdk

-真机yum源配置

mkdir /var/ftp/elk

scp /linux\_soft/04/elk/\*.rpm /var/ftp/elk/

cd /var/ftp/elk

createrepo .

-es1YUM仓库配置

......

[elk\_repo]

name=CentOS-elk

baseurl="ftp://192.168.1.254/elk"

enabled=1

gpgcheck=0

-安装软件包

yum -y install elasticsearch

-修改配置文件

/etc/elasticsearch/elasticsearch.yml

network.host 0.0.0.0

-启动服务

systemctl enable elasticsearch

systemctl start elasticsearch

-检查服务

netstat -ntulp | grep :9200

netstat -ntulp | grep :9300

-通过浏览器访问9200端口

curl http://192.168.1.51:9200

ES集群配置

-ES集群配置也很简单，只需要对配置文件做少量得修改即可，其它步骤和单机完全一致

-ES集群配置文件

cluster.name: myelk

node.name: es1

network.host: 0.0.0.0

discovery.zen.ping.unicast.hosts: [ "es1", "es2", "es3"]

ES集群配置

-集群中得所有节点要相互能够ping通，要在所有集群机器上配置/etc/hosts中得主机名与IP对应关系

/etc/hosts

-集群中所有机器都要安装java环境

-cluster.name集群名称配置要求完全一致

-node.name为当前节点标识，应配置本机得主机名

-discovery为集群节点机器，不需要全部配置

-配置完成以后启动所有节点服务

192.168.1.52-55

ansible-playbook一键部署

vim essetup.yml

---

- hosts: es

remote\_user: root

tasks:

- copy:

src: local.repo

dest: /etc/yum.repos.d/local.repo

owner: root

group: root

mode: 0644

- name: install elasticsearch

yum:

name: java-1.8.0-openjdk,elasticsearch

state: installed

- template:

src: elasticsearch.yml

dest: /etc/elasticsearch/elasticsearch.yml

owner: root

group: root

mode: 0644

notify: reload elasticsearch

tags: esconf

- service:

name: elasticsearch

enabled: yes

handlers:

- name: reload elasticsearch

service:

name: elasticsearch

state: restarted

vim hosts

::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

192.168.1.51 es1

192.168.1.52 es2

192.168.1.53 es3

192.168.1.54 es4

192.168.1.55 es5

vim ansible.cfg

defaults]

inventory = myhost

host\_key\_checking = False

vim myhost

[es]

192.168.1.[51:55]

[es:vars]

ansible\_ssh\_pass="a"

vim local.repo

[local\_repo]

name=CentOS-$releasever - Base

baseurl="ftp://192.168.1.254/centos-1804"

enabled=1

gpgcheck=1

[elk\_repo]

name=CentOS-elk

baseurl="ftp://192.168.1.254/elk"

enabled=1

gpgcheck=0

ansible批量执行

ansible es -m copy -a 'src=hosts dest=/etc/hosts'

ansible-playbook essetup.yml

验证集群

使用ES内置字段 \_cluster/health

curl http://192.168.1.51:9200/\_cluster/health?pretty

ES插件的使用

head插件

它展现ES集群的拓扑结构，并且可以通过它来进行索引(Index)和节点(Node)级别的操作

它提供一组针对集群的查询API，并将结果以json和表格形式返回

它提供一些快捷菜单，用以展现集群的各种状态

kopf插件

是一个Elasticsearch的集群监控工具

它提供了对ES集群操作的API

bigdesk插件

可以通过它来查看ES集群的各种状态，如：cpu，内存使用情况，索引数据，搜索情况，http连接数等

ES插件

查看

/usr/share/elasticsearch/bin/plugin list

安装

/usrshare/elasticsearch/bin/plugin install ftp://192.168.1.254/elk/elasticsearch-head-master.zip

这里必须使用url的方式进行安装，如果文件在本地，我们也需要使用file://的方式指定路径，例如文件在/tmp/xxx下面

我们要写成 file:///tmp/xxx 删除使用remove指令

扩展插件

RESTful API

http请求由三部分组成

分别是： 请求行，消息报头，请求正文

请求行以一个方法符号开头，以空格分开，后面跟着请求的URI和协议版本，格式如下：

Method Request-URI HTTP-Version CRLF

http请求方法

常用方法 GET POST HEAD

其它方法 OPTIONS PUT DELETE TRACE 和 CONNECT

ES常用

PUT 增

DELET 删

POST 改

GET 查

系统命令curl

在linux中curl是一个利用URL规则在命令行下工作的文件传输工具，可以说是一款很强大的http命令行工具

它支持多种模式，自定义请求头等强大功能，是一款综合工具

curl常用参数介绍

-A 修改请求agent

-X 设置请求方法

-i 显示返回头信息

RESTful API 调用

Elasticsearch提供了一系列RESTful的API

检查集群，节点，索引的健康度，状态和统计

管理集群，节点，索引的数据及元数据

对索引进行CRUD操作及查询操作

执行其他高级操作如分页，排序，过滤等

POST或PUT数据使用json格式

json是javaScript对象表示法，它是一种基于文本独立于语言的轻量级书记交换格式

JSON格式

数组 ["aa","bb","cc"]

键值对 { key： value}

RESTful API 简单使用

\_cat API查询集群状态，节点信息

查看帮助信息

http://192.168.1.51:9200/\_cat

v参数显示详细信息

http://192.168.1.51:9200/\_cat/health?v

help显示帮助信息

http://192.168.1.51:9200/\_cat/health?help

分片信息

http://192.168.1.51:9200/\_cat/shards?v

nodes 查询节点状态信息

http://192.168.1.51:9200/\_cat/nodes?v

索引信息

http://192.168.1.51:9200/\_cat/indices?v

RESTful API增加

创建一个索引，并设置分片数量与副本数量

curl -XPUT 'http://192.168.1.55:9200/tedu/' -d '{

"settings":{

"index":{

"number\_of\_shards":5,

"number\_of\_replicas":1

}

}

}'

RESTful API插入数据

(增)加数据，使用PUT方法

调用方式:数据库地址/索引/类型/id值

curl -XPUT 'http://192.168.1.55:9200/tedu/teacher/1' -d '{

"职业":"诗人",

"名字":"李白",

"称号":"诗仙",

"年代":"唐"

}' 休息到 21：20

POST 修改

修改数据，使用POST方法

在修改数据的时候必须调用 \_update 关键字

调用方式： 数据库地址/索引/类型/id值/\_update

curl -XPOST

'http://192.168.1.55:9200/tedu/teacher/3/\_update' -d {

"doc":{

"年代":"唐代"

}

}'

查询

查询使用GET方法，GET为默认方法

查询显示结果时候可以用pretty规范显示格式

多条查询需要使用\_mget 关键字配合 json 调用

curl -XGET 'http://192.168.1.55:9200/tedu/teacher/1'

删除

删除时候可以是文档，也可以是库，但不能是类型

curl -XDELETE 'http://192.168.1.55:9200/tedu/teacher/1'

curl -XDELETE 'http://192.168.1.55:9200/tedu'

批量导入数据

使用\_bulk批量导入数据

gzip -d logs.jsonl.gz

批量导入使用POST方式，数据格式为json，编码使用--data-binary

curl -XPOST http://192.168.1.51:9200/\_bulk --data-binary @logs.jsonl

当要导入的json格式数据中没有指定类型，那么导入时需要指定索引，类型

curl -XPOST http://192.168.1.51:9200/oo/xx/\_bulk --data-binary @accounts.json

Kibana

数据可视化平台

特点：

灵活的分析和可视化平台

kibana安装

创建新虚拟机 base-vm kibana （192.168.1.56）

yum -y install kibana

kibana默认安装目录在 /opt/kibana下面，配置文件在 /opt/kibana/condig/kibana.yml

修改配置文件

server.port: 5601

server.host: "0.0.0.0"

elasticsearch.url: "es1:9200"

kibana.index: ".kibana"

kibana.defaultAppId: "discover"

elasticsearch.pingTimeout: 1500

elasticsearch.requestTimeout: 30000

elasticsearch.startupTimeout: 5000

启动服务

systemctl enable kibana

systemctl start kibana

Logstash

数据采集，加工处理以及传输的工具

特点：

所有类型的数据集中处理

不同模式和格式数据的正常化

自定义日志格式的迅速扩展

为自定义数据源轻松添加插件

logstash安装

创建新的虚拟机：4Gmem 2Cpu IP：1.57 Hostname：logstash

logstash依赖Java环境，需要安装java-1.8.0-openjdk

yum -y install java-1.8.0-openjdk

logstash没有默认的配置文件，需要手动配置

logstash安装在/opt/logstash目录下

yum -y install logstash

logstash工作结构

{数据源} ==>

input{ } ==>

filter{ } ==>

output{ } ==>

{ ES }

logstash类型及条件判断

logstash里面的类型

布尔值类型: ssl\_enable => true

字节类型： bytes => "1MiB"

字符串类型： name => "xkops"

数值类型： port => 22

数组： match => ["datetime","UNIX"]

哈希： options => {k => "v" , k2=> "v2"}

编码解码： codec => "json"

路径： file\_path => "/tmp/filename"

注释： #

logstash条件判断

等于： ==

不等于： !=

小于： <

大于： >

小于等于： <=

大于等于： >=

匹配正则： =~

不匹配正则： !~

logstash条件判断

logstash条件判断

包含： in

不包含： not in

与： and

或： or

非与： nand

非或： xor

复合表达式： ()

取反符合： !()

logstash的第一个配置文件

/etc/logstash.conf

input{

stdin{}

}

filter{ }

output{

stdout{}

}

启动并验证

logstash -f logstash.conf 休息15分钟 21：15

logstash插件

上页的配置文件使用了logstash-input-stdin和logstash-output-stdout两个插件，logstash还由filter和codec类插件

查看插件的方式是

/opt/logstash/bin/logstash-plugin list

插件及文档地址

https://github.com/logstash-plugins

codec类插件

常用的插件：plain，json，json\_lines,rubydebug,multiline等

使用上面的例子，这次输入json数据

设置输入源的codec是json，在输入的时候选择rubydebug

input{

stdin{ codec => "json" }

}

filter{ }

output{

stdout{ codec => "rubydebug"}

}

输入普通数据和json对比

{"a":1,"c":3,"b":2}

apache日志格式

192.168.1.57 - - [28/Apr/2020:21:26:29 +0800] "GET / HTTP/1.1" 200 12 "-" "curl/7.29.0"

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined

input file 插件

input{

file {

path => ["/tmp/a.log","/tmp/b.log"]

}

}

filter{ }

output{

stdout{

codec => rubydebug }

}

创建日志文件 touch /tmp/{a,b}.log

输入日志并验证 echo A\_${RANDOM} >> /tmp/a.log

echo B\_${RANDOM} >> /tmp/b.log

/opt/logstash/bin/logstash -f /etc/logstash/logstash.conf

练习input file 插件

删除原有指针文件：rm -rf .sincedb\_ab3977c541d1144f701eedeb3af4956a

file{

start\_positon => "beginning"

sincedb\_path => "/var/lib/logstash/sincedb-access"

path => ["/tmp/a.log" , "/tmp/b.log"]

type => 'filelog'

}

sincedb\_path 记录读取文件的位置

start\_position配置第一次读取文件从什么地方开始

filter grok插件

解析各种非结构化的日志数据插件

grok使用正则表达式把非结构化的数据结构化

在分组匹配，正则表达式需要根据具体数据结构编写

虽然编写困难，但适用性极广

几乎可以应用于各类数据

grok{

match => ["message",(?<key>reg), "%{IP:clientip}"]

}

grok正则分组匹配

匹配IP时间戳和请求方法

match => {"message" => "(?<ip>[0-9.]+)(?<ident>[0-9a-z\-]+)(?<time>.+)\]\"(?<verb>[A-Z]+)(?<url>)\S+(?<proto>[A-Z]+)\/(?<ver>[0-9.]+)\"(?<rc>\d+)(?<siza>\d+)\"(?<ref>\S+)\"\"(?<agent>[^\"]+)\""}

使用正则宏 /opt/logstash/vendor/bundle/jruby/1.9/gems/logstash-patterns-core-2.0.5/patterns/grok-patterns

match => {"message" => "%[IP:client]" %{WORD:method}

%{URIPATHPARAM:request}%{NUMBER:bytes}

%{NUMBER:dutation}"}

最终版本

match => {"message" => "%{COMBINEDAPACHELOG}"}

百度搜索：《正则表达式必知必会》

output ES插件

if[type] == "apache\_log"{

elasticsearch {

hosts =>["192.168.1.51:9200" , "192.168.1.52:9200"]

index => "filelog"

flush\_size => 2000

idle\_flush\_time => 10

}}

调试成功后，把数据写入ES集群 休息带实验到 21：20

input filebeats插件

beats {

port => 5044

}

这个插件主要用来接收beats类软件发送过来的数据，由于logstash依赖JAVA环境，而且占用资源非常大，因此会使用更轻量的filebeat替代。

filebeat安装与配置

使用rpm安装filebeat

yum -y install filebeat

修改配置文件/etc/filebeat/filebeat.yml

设置开机运行

systemctl enable filebeat

开启服务

systemctl start filebeat

filebeat安装配置

修改配置文件 /etc/filebeat/filebeat.yml

paths:

- /var/log/httpd/access\_log

document\_type: apache\_log

output:

logstash

hosts:["192.168.1.57:5044"]

vim /etc/logstash/logstash.conf

input{

file {

start\_position => "beginning"

sincedb\_path => "/var/lib/logstash/sincedb-access"

path => ["/tmp/a.log"]

type => "testlog"

}

beats {

port => 5044

}

}

filter{

if [type] == "apache\_log" {

grok {

match => { "message" => "%{COMBINEDAPACHELOG}" }

}

}

}

output{

if [type] == "apache\_log" {

elasticsearch {

hosts => ["es1:9200","es2:9200","es3:9200"]

index => "weblog"

flush\_size => 2000

idle\_flush\_time => 10

}}

}

大数据

企业利用相关数据分析帮助他们降低成本，提高效率，开发新产品，做出更明智得业务决策等。

把数据集合并后进行分析得出得信息和数据关系性，用来察觉商业趋势，判定研究质量，避免疾病扩散，打击犯罪或测定即时交通路况等

大规模并行处理数据库，数据挖掘电网，分布式文件系统或数据库，云计算和可扩展得存储系统等

大数据5V特性

(V)olume(大体量)

可从数百TB到数十数百PB，甚至EB的规模

(V)ariety(多样性)

大数据包括各种格式和形态的数据

(V)elocity(时效性)

很多大数据需要在一定的时间限度下得到及时处理

(V)elocity(准确性)

处理的结果要保证一定的准确性

(V)alue(大价值)

大数据包含很多深度的价值，大数据分析挖掘和利用将带来巨大的商业价值

大数据与Hadoop

Hadoop是一种分析和处理海量数据的软件平台

Hadoop是一款开源软件，使用JAVA开发

Hadoop可以提供一个分布式基础架构

Hadoop特点

高可靠性，高扩展性，高效性，高容错性，低成本

Hadoop常用组件

HDFS：Hadoop分布式文件系统（核心组件）

MapReduce：分布式计算框架（核心组件）

Yarn：集群资源管理系统（核心组件）

Zookeeper：分布式协作服务

Hbase：分布式列存数据库

Hive：基于Hadoop的数据仓库

Sqoop：数据同步工具

Pig：基于Hadoop的数据流系统

Mahout：数据挖掘算法库

Flume：日志收集工具

HDFS角色及概念

Hadoop体系中数据存储管理的基础，是一个高度容错系统，用于在低成本的通用硬件上运行。

角色和概念

-Client

-Namenode

-Secondarynode

-Datanode

NameNode

Master节点，管理HDFS的名称空间和数据块映射信息，配置副本策略，处理所有客户端请求

Secondary NameNode

定期合并fsimage 和 fsedits ，推送给NameNode

紧急情况下，可辅助恢复NameNode

但Secondary NameNode并非NameNode的热备

DataNode

数据存储节点，存储实际的数据

汇报存储信息给NameNode

Client

切分文件

访问HDFS

与NameNode交互，获取文件位置信息

与DataNode交互，读取和写入数据

Block

每块缺省128MB大小

每块可以多个副本

MapReduce角色及概念

源自于Google的MapReduce论文，JAVA实现的分布式计算框架

角色和概念

-JobTracker

-TaskTracker

-Map Task

-Reducer Task

JobTracker

Master节点只有一个

管理所有作业/任务的监控，错误处理等

将任务分解成一系列任务，并分派给Task Tracker

TaskTracker

Slave节点，一般是多台

运行MapTask和Reduce Task

并与JobTracker交互，汇报任务状态

Map Task

解析每条数据记录，传递给用户编写的map()并执行，将输出结果写入本地磁盘

单机模式hadoop安装

1.创建虚拟机nn01，IP：192.168.1.60

2.安装配置java环境，安装jps工具

yum -y install java-1.8.0-openjdk-devel

3.获取软件

scp 192.168.1.254:/linux-soft/04/hadoop/hadoop-2.7.7.tar.gz .

或者

http://hadoop.apache.org

4.加压hadoop-2.7.7.tar.gz

tar -xf hadoop-2.7.7.tar.gz

mv hadoop-2.7.7 /usr/local/hadoop

chown -R 0.0 /usr/local/hadoop

5.设置环境变量，启动运行

vim /usr/local/hadoop/etc/hadoop-env.sh

export JAVA\_HOME="/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-2.b14.el7.x86\_64/jre/"

export HADOOP\_CONF\_DIR="/usr/local/hadoop/etc/hadoop"

hadoop单机模式统计英文单词出现次数

vim /etc/hosts

192.168.1.60 nn01

cd /usr/local/hadoop

mkdir input

cp \*.txt input/

./bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.7.jar wordcount input output

Hadoop伪分布式

伪分布式的安装和完全分布式类似，区别是所有角色安装在一台机器上，使用本地磁盘，一般生产环境都会使用完全分布式，伪分布式一般用来学习和测试Hadoop的功能

配置文件

hadoop-env.sh 环境配置文件

core-site.xml Hadoop核心配置文件

hdfs-site.xml Hdfs组件配置文件

mapred-site.xml.template MapReduce分布式计算框架配置文件

yarn-site.xml Yarn集群资源管理系统

slaves 从节点配置文件

文件格式

Hadoop-env.sh

JAVA\_HOME

HADOOP\_CONF\_DIR

xml文件配置格式

<property>

<name>关键字</name>

<value>变量值</value>

<description>描述</description>

</property>

HDFS分布式文件系统（新的虚拟机安装Hadoop）

完全分布式

192.168.1.60 NameNode/SecondaryNameNode nn01

192.168.1.61 DataNode node1

192.168.1.62 DataNode node2

192.168.1.63 DataNode node3

主机名映射

/etc/hosts(所有虚拟机)

192.168.1.60 nn01

192.168.1.61 node1

192.168.1.62 node2

192.168.1.63 node3

基础环境准备

禁用selinux

SELINUX=disable

禁用firewalld

systemctl stop firewalld

systemctl mask firewalld

安装java-1.8.0-openjdk-devel

yum -y install java-1.8.0-openjdk-devel

配置SSH信任关系

不能出现要求输入yes的情况，每台机器都要连接成功

vim /etc/ssh/ssh\_config

Host \*

GSSAPIAuthentication yes

StrictHostKeyChecking no

配置免密钥登录

ssh-keygen -t rsa -f "/root/.ssh/id\_rsa" -N ''

for i in {60..63}

do

ssh-copy-id -i id\_rsa.pub 192.168.1.$i

done

核心配置文件

vim core-site.xml

<property>

<name>fs.defaultFS</name>

<value>hdfs://nn01:9000</value>

<description>描述</description>

</property>

<property>

<name>hadoop.tmp.dir</name> #所有数据的根目录

<value>/var/hadoop</value> #生产环境中单独给一块硬盘mount使用

<description>描述</description>

</property>

HDFS配置文件

vim hdfs-site.xml

<property>

<name>dfs.datanode.http.address</name>

<value>nn01:50075</value>

<description>声明Namenode地址</description>

</property>

<property>

<name>dfs.namenode.secondary.http-address</name>

<value>nn01:50090</value>

<description>声明SecondaryNameNode地址</description>

</property>

<property>

<name>dfs.replication</name>

<value>2</value>

<description>副本策略</description>

</property>

slaves配置文件

只写DateNode节点的主机名称

node1

node2

node3

同步配置

Hadoop所有节点的配置参数完全一样

for i in node{1..3}

do

scp /usr/local/hadoop/etc/hadoop/\* $i:/usr/local/hadoop/etc/hadoop/;

done

在本机创建/var/hadoop文件夹

mkdir /var/hadoop

在namenode上执行格式化操作

./bin/hdfs namenode -format

启动集群

./sbin/start-dfs.sh

验证集群

./bin/hdfs dfsadmin -report

Hadoop三大核心组件

HDFS分布式文件系统（已完成） MapReduce分布式计算框架 Yarn集群资源管理

MapReduce部署

mv mapred-site.xml.template mapred-site.xml

vim mapred-site.xml

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

<description>运行mapreduce时用谁来管理</description>

</property>

yarn部署

vim yarn-site.xml

<property>

<name>yarn.resourcemanager.hostname</name>

<value>nn01</value>

<description>声明在哪台机器上启动resourcemanager</description>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

<description>计算框架的名称</description>

</property>

配置文件同步到所有机器

for i in node{1..3}

do

rsync -aXSH --delete /usr/local/hadoop/etc ${i}:/usr/local/hadoop/

done

启动服务

/usr/local/hadoop/sbin/start-yarn.sh

验证服务

./bin/yarn node-list

使用web访问Hadoop

namenode Web页面（nn01）

http://192.168.1.60:50070/

secondory namenode Web页面（nn01）

http://192.168.1.60:50090/

datanode Web页面（node1，node2，node3）

http://192.168.1.61:50075/

resourcemanager Web页面（nn01）

http://192.168.1.60:8088/

nodemanager Web页面（node1，node2，node3）

http://192.168.1.61:8042/

HDFS使用

./bin/hadoop fs -ls / ---> 对应shell命令 ls /

./bin/hadoop fs -mkdir /abc ---> 对应shell命令 mkdir /abc

./bin/hadoop fs -touchz /urfile ---> 对应shell命令 touch /urfile

./bin/hadoop fs -put localfile /remotefile 上传文件

./bin/hadoop fs -get /remotefile 下载文件

报错处理

常见报错信息：Name node is in safe......

处理方法：执行命令 ./bin/hadoop dfsadmin -safemode leave

在hdfs文件创建文件夹，上传文件至hdfs文件系统中，使用hadoop分析文件系统中的单单词出现频率，展示结果。

./bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.7.jar wordcount /remotefile /output

准备两台虚拟机

192.168.1.64 newnode

192.168.1.65 nfsgw

安装 java-1.8.0-openjdk-devel

增加节点

启动一个新的系统，设置ssh免密码登录

在所有节点修改 /etc/hosts，增加新节点的主机信息

安装java运行环境（java-1.8.0-openjdk-devel）

修改NameNode的slaves文件增加该节点

拷贝NameNode的/usr/local/hadoop到本机

在该节点启动DATaNode

./sbin/hadoop-daemon.sh start datanode

设置同步带宽，并同步数据

./bin/hdfs dfsadmin -setBalancerBandwidth 600000000

./sbin/start-balancer.sh

查看集群状态

./bin/hdfs dfsadmin -report

修复节点与增加节点基本一致

启动服务

./sbin/hadoop-daemon.sh start datanode

数据恢复时自动的

上线以后会自动恢复数据，如果数据量非常巨大，可能需要一定的时间

HDFS删除节点

配置NameNode的hdfs-site.xml

增加dfs.hosts.exclude配置

<property>

<name>dfs.hosts.exclude</name>

<value>/usr/local/hadoop/exclude</value>

<description>声明要删除节点的主机名</description>

</property>

更新数据

./bin/hdfs dfsadmin -refreshNodes

yarn节点管理

由于Hadoop在2.x引入了yarn框架，对于计算节点的操作已经变的非常简单

增加节点

sbin/yarn-daemon.sh start nodemanager

删除节点

sbin/yarn-daemon.sh stop nodemanager

查看节点

./bin/yarn node -list

yarn不包含数据，所以在增加删除修复节点时比较简单

NFS网关

准备虚拟机两台

192.168.1.65 nfsgw 安装java-1.8.0-openjdk-devel

192.168.1.66 client

nfs网关用途

用户可以通过操作系统兼容本地NFSv3客户端来浏览HDFS文件系统

用户可以从HDFS文件系统下载文档到本地文件系统

用户可以通过挂载点直接流化数据，支持文件附加，但是不支持随机写

NFS网关支持NFSv3和允许HDFS作为客户端文件系统的一部分被挂载

nfs网关特性

HDFS超级用户时NameNode进程本身具有相同标识的用户，超级用户可以执行任何操作，因为权限检查永远不会认为超级用户失败

注意事项

在非安全模式下，运行网关进程的用户时代理用户

在安全模式下，KerberosKeytab的用户时代理用户

配置代理用户

在NameNode和NFSGW上添加代理用户

代理用户的UID，GID，用户名必须完全相同

如果因特殊原因客户端的用户和NFS网关的用户UID不能保持一致，需要我们配置nfs.map的静态映射关系

nfs.map

uid 10 100 # Map the remote UID 10 the local UID 100

gid 11 101 # Map the remote GID 10 to the local GID 101

配置用户

在namenode(nn01)上添加用户和组

groupadd -g 800 nfsuser

useradd -u 800 -g 800 -r -d /var/hadoop nfsuser

在nfs网关服务器也同样执行以上两条命令

配置core-site.xml

核心配置core-site.xml

hadoop.proxyuser.{代理用户}.groups

hadoop.proxyuser.{代理用户}.hosts

这里的{代理用户}是主机上真实运行的nfs3的用户

在非安全模式下，运行nfs网关的用户为代理用户

groups为挂载点用户所使用的组

hosts为挂载点主机地址

vim core-site.xml

<property>

<name>hadoop.proxyuser.nfsuser.groups</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.nfsuser.hosts</name>

<value>\*</value>

</property>

配置步骤

停止集群所有服务

./sbin/stop-all.sh

同步配置文件到所有主机

启动hdfs

./sbin/start-dfs.sh

NFSGW配置

配置步骤

启动一个新的系统，卸载rpcbind，nfs-utils

配置/etc/hosts,添加所有NameNode和DataNode的主机名与IP对应关系

安装JAVA运行环境（java-1.8.0-openjdk-devel）

同步NameNode的/usr/local/hadoop到本机

配置hdfs-site.xml

启动服务

配置hdfs-site.xml

nfs.exports.allowed.hosts

默认情况下，export可以被任何客户端挂载，为了更好的控制访问，可以设置属性。

值和字符串对应机器名和访问策略，通过空格来分割。机器名的格式可以是单一的主机，

java的正则表达式或者IPv4地址

使用rw或ro可以指定导出目录的读写或只读权限

如果访问策略没被提供，默认为只读。每个条目使用";"来分割

hdfs-site.xml配置

nfs.exports.alowed.hosts

配置 \* rw

<property>

<name>nfs.exports.alowed.hosts</name>

<value>\* rw</value>

</property>

nfs.dump.dir

用户需要更新文件转储目录参数。NFS客户端经常重新安排写操作，顺序的写操作会随机到达NFS网关。

这个目录常用户临时存储无序的写操作。对于每个文件，无序的写操作会在他们积累在内存中超过一定

阈值（如1M）时被转储。需要确保有足够的空间的目录

如：应用上传10个100M，那么这个转储目录推荐1GB左右空间，以便每个文件都发生最坏的情况。

只有NFS网关需要在设置该属性后重启

<property>

<name>nfs.dump.dir</name>

<value>/var/nfstmp</value>

</property>

NFS启动与挂载

设置/usr/local/hadoop/logs权限，为代理用户赋予读写执行的权限

setfacl -m nfsuser:rwx /usr/local/hadoop/logs

使用root用户启动portmap服务

./sbin/hadoop-daemon.sh --script ./bin/hdfs start portmap

使用代理用户启动nfs3

sudo -u nfsuser ./sbin/hadoop-daemon.sh --script ./bin/hdfs start nfs3

注意：

启动portmap需要使用root用户

启动nfs3需要使用core-site里面设置的代理用户

必须先启动portmap之后再启动nfs3

如果portmap重启了，再重启之后nfs3也需要重启

目前NFS只能使用V3版本 vers=3

仅使用TCP作为传输协议 proto=tcp

不支持NLM nolock

禁用access time的时间更新 noatime

禁用acl扩展权限 noacl

NFS启动与挂载

强烈建议使用安装选项sync，它可以最小化避免重排序写入造成不可预测的吞吐量，

未指定同步选项可能会导致上传大文件时出现不可靠行为

启动一台机器并安装nfs-utils

yum -y install nfs-utils

挂载nfs

mount -t nfs -o vers=3,proto=tcp,noatime,nolock,sync,nocal 192.168.1.65:/ /mnt/

实现开机自动挂载

vim /etc/fstab

192.168.1.65:/ /mnt/ nfs vers=3,proto=tcp,nolock,noatime,sync,noacl,\_netdev 0 0

mount -a

Hadoop常用组件Zookeeper

zookeeper是一个开源的分布式应用程序协调服务

zookeeper是用来保证数据再集群间的事物一致性

应用场景

集群分布式锁

集群统一命名服务

分布式协调服务

角色与特性

leader：接受所有Follower的提案请求并统一协调发起提案的投票，负责与所有的Follower进行内部数据交换

Follower：直接为客户端服务并参与提案的投票，同时与leader进行数据交换

Observer：直接为客户端服务但并不参与提案的投票，同时也与leader进行数据交换

角色与选举

服务在启动的时候是没有角色的（LOOKING）

角色是通过选举产生的

选举产生一个Leader，剩下的是Follower

选举Leader原则

集群中超过半数机器投票选择Leader

假如集群中拥有n台服务器，那么leader必须得到n/2+1台服务器得投票

如果leader死亡，重新选举leader

如果死亡得机器数量达到一半，则集群挂掉

如果无法得到足够得投票，就重新发起投票，如果参与投票得机器不足n/2+1，则集群停止工作

群也会停止工作

Observer不计算在投票总设备数量里面

Zookeeper可伸缩扩展性原理与设计

Leader所有写相关操作

Follower读操作与响应leader提议

在observer出现以前，zookeeper得伸缩性由follower来实现，我们可以通过添加follower节点得数量来保证zookeeper服务得读性能，

但是随着follower节点数量得增加，zookeeper服务得写性能受到了影响

客户端提交一个请求，若是读请求，则由每台server得本地副本数据库直接响应，若是写请求，需要通过一致性协议（Zab）来处理

Zab协议规定：来自Client得所有写请求都要转发给ZK服务中唯一得leader，由leader根据该请求发起一个Proposal。然后其他得Server对该

Proposal进行Vote，之后leader对Vote进行收集，当vote数量过半时leader会向所有得Server发送一个通知消息，最后当client所连接得server收到

该消息时，会把该操作更新到内存中并对client得写请求做出回应

zookeeper在上述协议中实际扮演了两个职能，一方面从客户端接受连接与操作请求，另一方面对操作结果进行投票，这两个职能在Zookeeper集群扩展得时候彼此制约

从Zab协议对写请求得处理过程中可以发现，增加Follower得数量，则增加了协议投票过程得压力，因为leader节点必须等待集群中过半Server响应投票，是节点得增加

使得部分计算机运行较慢，从而拖慢整个投票过程得可能性也随之提高，随着集群变大，写操作也会随之下降

所以，我们不得不在增加client数量得期望和我们希望保持较好吞吐性能得期望间进行权衡，要打破这一耦合关系，我们引入了不参与投票得服务器observer。

observer可以接受客户端得连接，并将写请求转发给leader节点，但leader节点不会要求observer参加投票，仅仅在上述第3步那样，和其它服务节点一起得到投票结果

observer得扩展，给zookeeper得可伸缩性带来了全新得镜像，加入很多observer节点，无须担心严重影响吞吐量，但并非无懈可击，因为协议中得通知阶段，仍然与服务器

得数量呈线性关系。但是这里得串行开销非常低，因此，我们可以认为在通知服务器阶段得开销不会成为瓶颈

observer提升读性能得可伸缩性

observer提供了广域网能力

zookeeper集群得安装配置

配置文件改名zoo.cfg

mv zoo\_sample.cfg zoocfg

zoo.cfg文件最后添加如下内容

server.1=node1:2888:3888

server.2=node2:2888:3888

server.3=node3:2888:3888

server.4=nn01:2888:3888:observer

同步配置文件到所有节点

for i in node{1..3}

do

rsync -aXSH /usr/local/zookeeper/ ${i}:/usr/local/zookeeper/

done

zoo.cfg集群得安装配置

创建datadir指定的目录

mkdir /tmp/zookeeper

在目录下创建id对应主机名的myid文件

所有节点配置数据目录，并配置myid文件

关于myid文件

myid文件中只有一个数字

注意：请确保每个server的myid文件中id数字不同

server.id中的id与myid中id必须一致

id的范围是1~255.

for i in {1..3}; do ssh node$i "echo $i > /tmp/zookeeper/myid"; done

验证配置：for i in node{1..3}; do ssh $i "cat /tmp/zookeeper/myid"; done

启动集群，查看验证（在所有集群节点执行）

/usr/local/zookeeper/bin/zkServer.sh start

查看角色

/usr/local/zookeeper/bin/zkServer.sh status

Zookeeper管理文档

http://zookeeper.apache.org/doc/r3.4.10/zookeeperAdmin.html

脚本管理

#!/bin/bash

exec 7<>/dev/tcp/$1/2181 #打开文件描述符

echo "ruok" >&7 #发送指令

read xx <&7 #读取指令结果

echo ${xx} #输出结果

exec 7<&- #关闭文件描述符

Kafka集群

Kafka是什么

Kafka是由LinkedIn开发的一个分布式的消息系统

Kafka是使用Scala编写

Kafka是一种消息中间件

为什么要使用Kafka

解耦，冗余，提高扩展性，缓冲

保证顺序，灵活，削峰填谷

异步通信

Kafka角色与集群结构

priducer：生产者，负责发部消息

consumer：消费者，负责读取处理消息

topic：消息的类别

Parition：每个topic包含一个或多个Partition

Broker：Kafka集群包含一个或多个服务器

Kafka通过Zookeeper管理集群配置，选举Leader