# AbutionGraph-分布式内存版（功能试用版）安装

AbutionGraph具有Graph（优先生产环境）、TmpGraph、MemoryGraph、ProxyGraph4种运行模式。

**1. 临时图TmpGraph的使用**

将开发依赖目录导入IDEA中即可运行示例程序（GraphOfTheGodsFactory.java），查看效果。

TmpGraph是利用JVM搭载的临时缓存，程序执行完毕则释放空间。

除了嵌入式的运行代码程序使用方式外，调试数据还可以在java11的REPL中（启动命名：abution-jshell）进行类似Python的代码调试。

**2. 内存图MemoryGraph的使用**

MemoryGraph是利用分布式缓存技术搭载的图谱存储系统，服务器端不停止则数据持续存储，否则数据清除并释放空间。

MemoryGraph可以将服务器端放在远程，通过IDEA编写客户端程序实时调试你的数据，这对于开发任务比较友好。

在Abution安装目录启动服务端：

java -Djava.ext.dirs=./lib/:./lib/ext/:./lib/spring/ -jar ./lib/MemoryAbutionGraph.jar

java -Djava.ext.dirs=./lib/:./lib/ext/:./lib/spring/ -jar ./lib/abution-start-2.1.0.jar

客户端使用1（本地IDEA软件执行）：

详见示例MemoryAbutionGraphExample.java。

执行示例程序后，图谱可视化页面中刷新选图，即可看到新增数据。

客户端使用2（用安装版自带的abution-jshell或java11的REPL命令行执行）：

$ abution-jshell (服务器执行该命令后粘贴代码执行)

可视化页面：

http://localhost:9995/graph/index.html

（目前版本仅图谱可视化可用）

**3. Graph实例的使用**

详见以下安装文档。

这是永久数据存储的模式，具有高可用特性。

# AbutionGraph-完整版安装

清空缓存命令：echo 1 > /proc/sys/vm/drop\_caches

AbutionGraph-2.8 集成软件参考版本信息：

基础组件:

Jdk-1.8.0\_211 (Abution-jshell使用自带jdk11)

Hadoop-3.3.5/Hadoop-2.7.7

Zookeeper-3.4.13（默认包含在Abution安装包中）

大数据生态组件（推荐版本）:

Python-3.8.5

Scala-2.12.8

Spark-3.3.2 （AI算法计算引擎；AbutionGCS-图计算定制版；Mlep可融合Python）

MySQL-8.0

Flink-1.12.1（实时计算引擎）

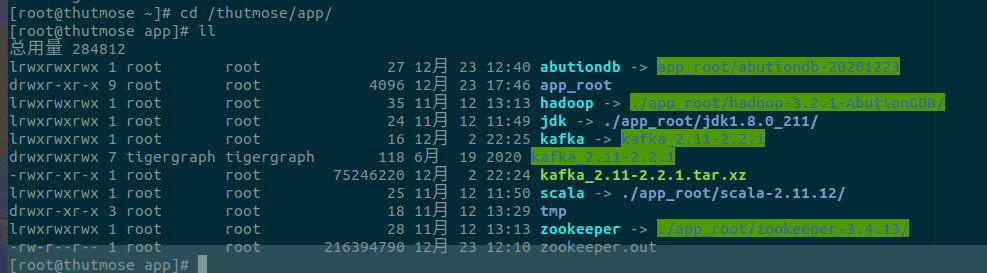
Hive-3.1.3（数据仓库API，推荐配置运行在Spark上）

RocketMQ-4.5.3

Kafka-2.2.1

Jcache

安装目录结构（JDK、Hadoop、AbutionDB），为了少改动系统环境变量，建议都装在/thutmose/app/（**mkdir -p /thutmose/app**）下：



安装步骤（按顺序操作执行）：

## 0）前期准备

**硬件**

最好是使用CentOS7或者Ubuntu18以上系统，不满足的话请升级系统内核gcc版本至8以上版本。由于我们实际上同时在整个群集中运行两个或三个系统：HDFS，AbutionDB和MapReduce，因此硬件通常由4至8个内核和8至32 GB RAM组成。这样一来，每个正在运行的进程至少可以具有一个内核，每个内核可以具有2-4 GB，提高吞吐和速度只需配置进程使用更多的资源。

一个运行HDFS的内核通常可以使2到4个磁盘繁忙，因此每台计算机通常可能只有2 x 300GB磁盘，最多有4 x 1TB或2TB磁盘。

AbutionDB的最低配置：如使用2H4GB的1u服务器，建议每台计算机最多运行两个进程-即DataNode和AbutionServer或DataNode和MR Worker，不建议同时启动3个服务。这个约束是为机器上的所有进程提供足够的可用堆空间。

**服务器设置**

1. 关闭防火墙；
2. （已安装则跳过）安装SSH无密通信：sudo apt-get install openssh-server，并执行：ssh-keygen -t rsa，一路回车即可，然后复制.ssh/id\_rsa.pub内容到.ssh/authorized\_keys中，最后启动服务：service ssh start；
3. （可选）在群集内运行NTP协议同步时间，时间不一致可能会导致节点间的数据同步延缓；
4. 如果是阿里等云服务器，请把端口号9090、50070、9990-9999 加入到安全组，否则web无法连通。

**规范化软件目录**

1. 执行命令新建安装目录：mkdir -p /thutmose/app/abution\_graph\_db\_install\_package/

2. 上传所有安装文件到：abution\_graph\_db\_install\_package/

3. 依次解压所有安装文件到文件夹abution\_graph\_db\_install\_package/下：tar -xvf xxx.tar.xz

4. 对所有软件目录创建软连接到/thutmose/app/（如上图目录结构）：

cd /thutmose/app/

ln -s ./abution\_graph\_db\_install\_package/jdk1.8.0\_211 ./jdk

ln -s ./abution\_graph\_db\_install\_package/scala-2.12.8 ./scala

ln -s ./abution\_graph\_db\_install\_package/hadoop-3.3.5 ./hadoop

ln -s ./abution\_graph\_db\_install\_package/abution-graph-db-2.8.1 ./abution

ln -s ./abution\_graph\_db\_install\_package/hive-3.1.3 ./hive

ln -s ./abution\_graph\_db\_install\_package/spark-3.3.2 ./spark

## 1）配置系统环境变量

**追加以下内容到文件（vim /etc/profile）末尾：**

## Hadoop-3.x

export HADOOP\_HOME=/thutmose/app/hadoop

export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

export CLASSPATH=".:$JAVA\_HOME/lib:$CLASSPATH"

export HADOOP\_COMMON\_HOME=$HADOOP\_HOME

export HADOOP\_HDFS\_HOME=$HADOOP\_HOME

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

export HADOOP\_YARN\_HOME=$HADOOP\_HOME

export HADOOP\_INSTALL=$HADOOP\_HOME

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

export HADOOP\_LIBEXEC\_DIR=$HADOOP\_HOME/libexec

export JAVA\_LIBRARY\_PATH=$HADOOP\_HOME/lib/native:$JAVA\_LIBRARY\_PATH

export PATH=$PATH:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin

## 遇到权限问题解开注释

export HDFS\_DATANODE\_USER=root

export HDFS\_DATANODE\_SECURE\_USER=root

export HDFS\_SECONDARYNAMENODE\_USER=root

export HDFS\_NAMENODE\_USER=root

export HADOOP\_SHELL\_EXECNAME=root

export YARN\_RESOURCEMANAGER\_USER=root

export YARN\_NODEMANAGER\_USER=root

## AbutionDB

export ABUTION\_HOME=/thutmose/app/abution

export ABUTION\_CONF\_DIR=$ABUTION\_HOME/conf

export PATH=$ABUTION\_HOME/bin:$PATH

## Zppkeeper

export ZOOKEEPER\_HOME=$ABUTION\_HOME/zk

export PATH=${ZOOKEEPER\_HOME}/bin:$PATH

## java8

export JAVA\_HOME=/thutmose/app/jdk

export JRE\_HOME=${JAVA\_HOME}/jre

export CLASSPATH=.:${JAVA\_HOME}/lib/dt.jar:${JAVA\_HOME}/lib/tools.jar

export PATH=${JAVA\_HOME}/bin:${JRE\_HOME}/bin:$PATH

## scala-2.12.8

export SCALA\_HOME=/thutmose/app/scala

export PATH=${SCALA\_HOME}/bin:$PATH

export TERM=xterm-color

## Hive

export HIVE\_HOME=/thutmose/app/hive

export PATH=${HIVE\_HOME}/bin:$PATH

## spark

export SPARK\_HOME=/thutmose/app/spark

export PATH=${SPARK\_HOME}/bin:$PATH

#export PYSPARK\_PYTHONPATH=${SPARK\_HOME}/bin:${SPARK\_HOME}/python:${SPARK\_HOME}/python/lib/py4j-0.10.7-src.zip:$PATH

## 配置Hadoop（单机可跳过）

注：Hadoop的配置可自行决定，与开源版本一致。

1. 进入目录：cd /thutmose/app/hadoop/etc/hadoop/
2. 配置JAVA\_HOME（默认已配好）：vim hadoop-env.sh
3. 更改localhost为主机名（可选,单机建议无需改动）

**vim core-site.xml**

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

**vim hdfs-site.xml**

<property>

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

<value>localhost:50070</value>

</property>

**vim workers**

localhost

**vim yarn-site.xml（可选-需要YARN时配置）**

<property>

<name>yarn.resourcemanager.address</name>

<value>localhost:8032</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>localhost:8030</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>localhost:8031</value>

</property>

1. 格式化HDFS：hdfs namenode -format -clusterid AbutionGraph
2. 启动hdfs：start-dfs.sh
3. 查看hdfs服务是否启动成功：jps

正常启动会有3个服务（NameNode、DataNode、SecondaryNameNode），否则检查log排除异常

hdfs监控页面：ip:50070

## 安装AbutionGraph

注：我们假定HDFS已经正常运行（单机版可不用，伪分布式可启用），具体在abution.properties中配置，。

1. 进入目录：cd /thutmose/app/abution/conf
2. 更改以下5个文件中localhost为主机名（可选,单机建议无需改动）

：gc、masters、monitor、slaves、tracers；高可用和多计算节点可依据服务器数量配置为多行的不同主机名。

1. 更改数据库配置文件abution.properties
2. instance.volumes：更改localhost为主机名（可选,单机建议无需改动）
3. 配置数据库为本地单机模式（如下）或分布式模式（改为使用hdfs目录，推荐单机模式也使用hdfs持久化数据）

## 设置数据持久化HDFS默认路径

#instance.volumes=hdfs://localhost:9000/abution

instance.volumes=file:///thutmose/app/abution/data#这里因为data内已经有了数据，在此路径初始化会报错，故改为test1文件夹

1. tserver.memory.maps.native.enabled：

建议设置true，可以解决JavaGC暂停的难题，提高AbutionDB性能，同时修改lib/libabution.so文件为对应服务器版本。

其中关联参数tserver.memory.maps.max和table.compaction.minor.logs.threshold用于调节吞吐量。

1. 启动Zookeeper：

cd /thutmose/app/abution/zk/bin

bash zkServer.sh start

1. 初始化Abution：

Source /etc/profile

abution init --instance-name [AbutionGraph] --user root --password [abutiongraph]

1. 启动Abution：abution-db-start.sh

jps查看是否启动成功，正常启动会有>5个Main服务，否则检查abution/logs排除异常。

AbutionDB监控页面：ip:9995

AbutionGraph-UI页面：ip:9995/graph

Abution起停命令：abution-cluster stop/start/restart

停止zk：zkServer.sh stop （启动Abution前必须先启动zk）

# AbutionGraph命令

停止AbutionGraph集群：

1. abution-rest-stop.sh（若几秒后还存在Bootstrap进程，请kill掉）
2. abution-db-stop.sh
3. zkServer.sh stop（可选）

启动AbutionGraph集群：

1. zkServer.sh start （已启动请忽略）
2. abution-db-start.sh
3. abution-rest-start.sh

HDFS启动/停止：start-dfs.sh / stop-dfs.sh

添加AbutionDB节点（在某个节点服务器中执行）：start-here.sh （需$ABUTION\_HOME/conf/slaves中已具有该节点）

停止AbutionDB节点（在某个节点服务器中执行）：stop-here.sh

清空缓存命令：echo 1 > /proc/sys/vm/drop\_caches

更多运维命令略。。。

# Hive on Abution storage

Hive是行式存储，检索效率底，但API很完善，拥有极大量的用户。Abution是列式存储，检索效率高，key-value的储存形式对于开发人员来说比较灵活，也意味着存储结构需要更深入的设计。而Hive-Abution-Storage则是将Abution用作Hive的数据存储，使用Hive的API来操作Abution，达到列式存储拥有表格存储的使用效果，并且达到列存的高效查询检索性能。

**MySQL-8的安装**

1. 下载MySQL

$ sudo apt install mysql-server

1. 配置MySQL

use mysql;

select user,host from user;

update user set authentication\_string='' where user='raini'; --将字段置为空，注意修改用户名'raini'为自己的

ALTER user 'root'@'localhost' IDENTIFIED BY 'abution-hive' WITH GRANT OPTION; --修改密码为abution-hive

flush privileges;

**Hive的安装（默认已配置好，重点配置MySQL的用户名秘密-高亮部分）**

1. 配置环境变量

export HIVE\_HOME=/thutmose/app/hive

export PATH=$PATH:$HIVE\_HOME/bin

1. 配置hive-env.sh

HADOOP\_HOME=/home/raini/app/hadoop

export HIVE\_CONF\_DIR=/home/raini/app/hive/conf

1. 配置日志

只需将conf目录中的hive-log4j2.properties.template复制一份修改名称为 hive-log4j2.properties 即可

1. 配置hive\_site.xml

增加mysql的连接地址和用户名密码，然后把mysql的驱动下载到 HIVE\_HOME/lib目录下。

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<!--JDBC元数据仓库连接字符串-->

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://thutmose:3306/metastore\_db?createDatabaseIfNotExist=true&amp;useSSL=false</value>

<description>JDBC connect string for a JDBC metastore</description>

</property>

<!--JDBC元数据仓库驱动类名-->

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

<description>Driver class name for a JDBC metastore</description>

</property>

<!--元数据仓库用户名-->

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

<description>Username to use against metastore database</description>

</property>

<!--元数据仓库密码-->

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>abution-hive</value>

<description>password to use against metastore database</description>

</property>

</configuration>

1. 复制MySQL依赖mysql-connector-java-8.0.13.jar到hive/lib/下
2. 初始化元数据库

schematool -dbType mysql -initSchema --verbose

注：如果想要急速的查询响应，可配置Hive执行引擎为Spark。

**Hive开发文档**（https://hive.apache.org/、https://www.docs4dev.com/docs/zh/apache-hive/3.1.1/reference/）

一些设置sql：

set hive.exec.dynamici.partition=true; #开启动态分区，默认是false

set hive.exec.dynamic.partition.mode=nonstrict; #开启允许所有分区都是动态的，否则须要有静态分区才能使用。

set hive.exec.dynamic.partition=true;(可通过这个语句查看：set hive.exec.dynamic.partition;)

set hive.exec.dynamic.partition.mode=nonstrict;

set hive.exec.max.dynamic.partitions=100000;(如果自动分区数大于这个参数，将会报错)

set hive.exec.max.dynamic.partitions.pernode=100000;

测试使用：

1. 启动Hive命令：

（用户名和密码查看abution/conf/abution.propties）

hive -hiveconf abution.instance.name=AbutionGraph -hiveconf abution.zookeepers=localhost -hiveconf abution.user.name=root -hiveconf abution.user.pass=abutiongraph

2.建表：

CREATE TABLE abution\_table(rowid STRING, name STRING, age INT, weight DOUBLE, height INT)  
 STORED BY 'org.apache.hadoop.hive.abution.AbutionStorageHandler'  
WITH SERDEPROPERTIES('abution.columns.mapping' = ':rowid, person:name, person:age, person:weight, person:height');

3.插入数据：

insert into table abution\_table values('ADB', 'AbutionGrapn', 5, 1.0, 1);

4.查询数据：

select \* from abution\_table;