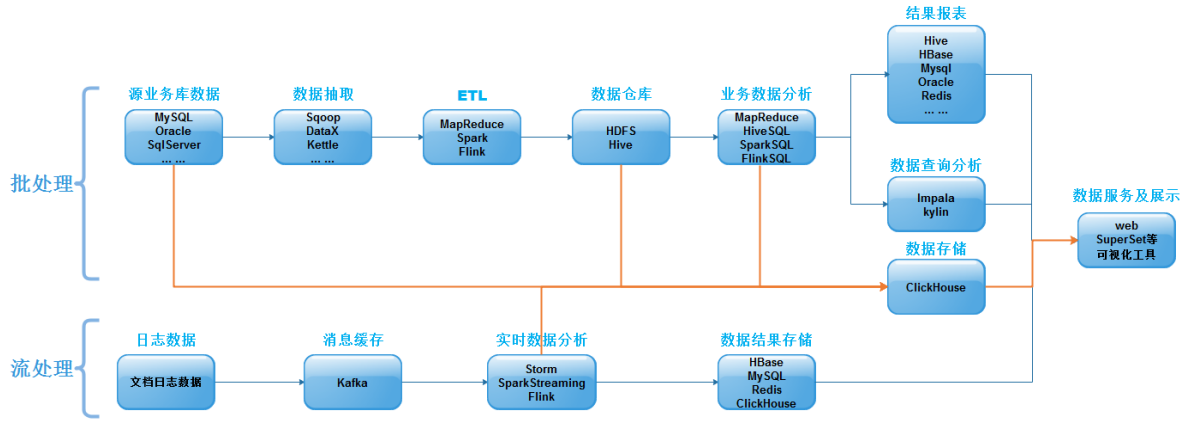
1.1 ClickHouse与其特性

在大数据处理场景中，流处理和批处理使用到的技术大致如下：



批处理会将源业务系统中的数据通过数据抽取工具（例如Sqoop）将数据抽取到HDFS中，这个过程可以使用MapReduce、Spark、Flink技术对数据进行ETL清洗处理，也可以直接将数据抽取到Hive数仓中，一般可以将结构化的数据直接抽取到Hive数据仓库中，然后使用HiveSQL或者SparkSQL进行业务指标分析，如果涉及到的分析业务非常复杂，可以使用Hive的自定义函数或者Spark、Flink进行复杂分析，这就是我们通常说的数据指标分析。分析之后的结果可以保存到Hive、HBase、MySQL、Redis等，供后续查询使用。一般在数仓构建中，如果指标存入Hive中，我们可以使用Sqoop工具将结果导入到关系型数据库中供后续查询。HBase中更擅长存储原子性非聚合查询数据，如果有大量结果数据后期不需要聚合查询，也可以通过业务分析处理考虑存入HBase中。对于一些查询需求结果反馈非常快的场景可以考虑将结果存入Redis中。

对于大多数企业构建数仓之后，会将结果存入到Hive中的DM层中。DM层数据存入的是与业务强相关的报表数据，DM层数据是由数仓中DWS层主题宽表聚合统计得到，这种报表层设计适合查询固定的场景。对于一些查询需求多变场景，我们也可以使用impala来直接将主题宽表数据基于内存进行交互式查询，对web或者数据分析做到交互式返回结果，使用impala对内存开销非常大。还有另外一种方式是使用Kylin进行预计算，将结果提前计算好存入Hbase中，以供后续交互式查询结果，Kylin是使用空间获取时间的一种方式，预先将各种维度组合对应的度量计算出来存入HBase,用户写SQL交互式查询的是HBase中预计算好的结果数据。最后将数据分析结果可以直接对web以接口服务提供使用或者公司内部使用可视化工具展示使用。

以上无论批处理过程还是流处理过程，使用到的技术几乎离不开Hadoop生态圈。

1.1.1 什么是ClickHouse

ClickHouse是一个开源的，用于联机分析（OLAP）的列式数据库管理系统（DBMS-database manager system），它是面向列的，并允许使用SQL查询，实时生成分析报告。ClickHouse最初是一款名为Yandex.Metrica的产品，主要用于WEB流量分析。ClickHouse的全称是Click Stream，Data WareHouse，简称ClickHouse。

ClickHouse不是一个单一的数据库,它允许在运行时创建表和数据库，加载数据和运行查询，而无需重新配置和重新启动服务器。ClickHouse同时支持列式存储和数据压缩，这是对于一款高性能数据库来说是必不可少的特性。一个非常流行的观点认为，如果你想让查询变得更快，最简单且有效的方法是**减少数据扫描范围**和 **数据传输时的大小**，而列式存储和数据压缩就可以帮助我们实现上述两点，列式存储和数据压缩通常是伴生的，因为一般来说列式存储是数据压缩的前提。

1.1.2 OLAP场景的特征

- 绝大多数是读请求。

- 数据以相当大的批次(> 1000行)更新，而不是单行更新;或者根本没有更新。
- 已添加到数据库的数据不能修改。
- 对于读取，从数据库中提取相当多的行，但只提取列的一小部分。
- 宽表，即每个表包含着大量的列。
- 查询相对较少(通常每台服务器每秒查询数百次或更少)。
- 对于简单查询，允许延迟大约50毫秒。
- 列中的数据相对较小：数字和短字符串(例如，每个URL 60个字节)。
- 处理单个查询时需要高吞吐量(每台服务器每秒可达数十亿行)。
- 事务不是必须的。
- 对数据一致性要求低。有副本情况下，写入一个即可，后台自动同步。
- 每个查询有一个大表。除了他以外，其他的都很小。
- 查询结果明显小于源数据。换句话说，数据经过过滤或聚合，因此结果适合于单个服务器的RAM中。

通过以上OLAP场景分析特点很容易可以看出，OLAP场景与其他通常业务场景(例如,OLTP或K/V)有很大的不同， 因此想要使用OLTP或Key-Value数据库去高效的处理分析查询场景，并不是非常完美的适用方案。例如，使用OLAP数据库去处理分析请求通常要优于使用MongoDB或Redis去处理分析请求。

1.1.3 ClickHouse特性

1.1.3.1 完备的DBMS功能

ClickHouse是一个数据库管理系统，而不仅是一个数据库，作为数据库管理系统具备完备的管理功能：

- DDL(Data Definition Language-数据定义语言)：可以动态地创建、修改或删除数据库、表和视图，而无须重启服务。
- DML(Data Manipulation Language)：可以动态查询、插入、修改或删除数据。
- 分布式管理：提供集群模式，能够自动管理多个数据库节点。
- 权限控制：可以按照用户粒度设置数据库或者表的操作权限，保障数据的安全性。
- 数据备份与恢复：提供了数据备份导出与导入恢复机制，满足生产环境的要求。

1.1.3.2 列式存储

目前大数据存储有两种方案可以选择，行式存储(Row-Based)和列式存储(Column-Based)。

- **行式存储在数据写入和修改上具有优势。**

行存储的写入是一次完成的，如果这种写入建立在操作系统的文件系统上，可以保证写入过程的成功或者失败，可以保证数据的完整性。列式存储需要把一行记录拆分成单列保存，写入次数明显比行存储多(因为磁头调度次数多，而磁头调度是需要时间的，一般在1ms~10ms)，再加上磁头需要在盘片上移动和定位花费的时间，实际消耗更大。

数据修改实际上也是一次写入过程，不同的是，数据修改是对磁盘上的记录做删除标记。行存储是在指定位置写入一次，列存储是将磁盘定位到多个列上分别写入，这个过程仍是行存储的列数倍。

所以，行式存储在数据写入和修改上具有很大优势。

- **列式存储在数据读取和解析、分析数据上具有优势。**

数据读取时，行存储通常将一行数据完全读出，如果只需要其中几列数据的情况，就会存在冗余列，出于缩短处理时间的考量，消除冗余列的过程通常是在内存中进行的。列存储每次读取的数据是集合的一段或者全部，不存在冗余性问题。

列式存储中的每一列数据类型是相同的，不存在二义性问题，例如，某列类型为整型int,那么它的数据集合一定是整型数据，这种情况使数据解析变得十分容易。相比之下，行存储则要复杂得多，因为在一行记录中保存了多种类型的数据，数据解析需要在多种数据类型之间频繁转换，这个操作很消耗CPU，增加了解析的时间。

所以，列式存储在数据读取和解析数据做数据分析上更具优势。

综上所述，行存储的写入是一次性完成，消耗的时间比列存储少，并且能够保证数据的完整性，缺点是数据读取过程中会产生冗余数据，如果只有少量数据，此影响可以忽略，数量大可能会影响到数据的处理效率。列存储在写入效率、保证数据完整性上都不如行存储，它的优势是在读取过程，不会产生冗余数据，这对数据完整性要求不高的大数据处理领域比较重要。一般来说一个OLAP类型的查询可能需要访问几百万或者几十亿行的数据，但是OLAP分析时只是获取少数的列，对于这种场景列式数据库只需要读取对应的列即可，行式数据库需要读取所有的数据列，因此这种场景更适合列式数据库，可以大大提高OLAP数据分析的效率。ClickHouse就是一款使用列式存储的数据库，数据按列进行组织，属于同一列的数据会被保存在一起，列与列之间也会由不同的文件分别保存，在对OLAP场景分析时，效率很高。

1.1.3.3 数据压缩

为了使数据在传输上更小，处理起来更快，可以对数据进行压缩，ClickHouse默认使用LZ4算法压缩，数据总体压缩比可达8:1。

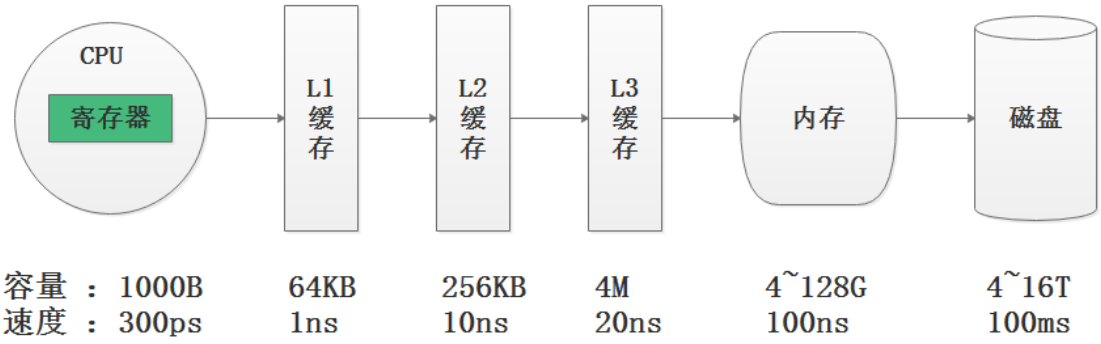
ClickHouse采用列式存储，列式存储相对于行式存储另一个优势就是对数据压缩的友好性。例如：有两个字符串“ABCDE”，“BCD”，现在对它们进行压缩：

```
压缩前：ABCDE_BCD
压缩后：ABCDE_(5,3)
```

通过以上案例可以看到，压缩的本质是按照一定步长对数据进行匹配扫描，当发现重复部分的时候就进行编码转换。例如：(5,3)代表从下划线往前数5个字节，会匹配上3个字节长度的重复项，即:“BCD”。当然，真实的压缩算法比以上举例更复杂，但压缩的本质就是如此，数据中重复性项越多，则压缩率越高，压缩率越高，则数据体量越小，而数据体量越小，则数据在网络中的传输越快，对网络带宽和磁盘IO的压力也就越小。

列式存储中同一个列的数据由于它们拥有相同的数据类型和现实语义，可能具备重复项的可能性更高，更利于数据的压缩。所以ClickHouse在数据压缩上比例很大。

1.1.3.4 向量化执行引擎



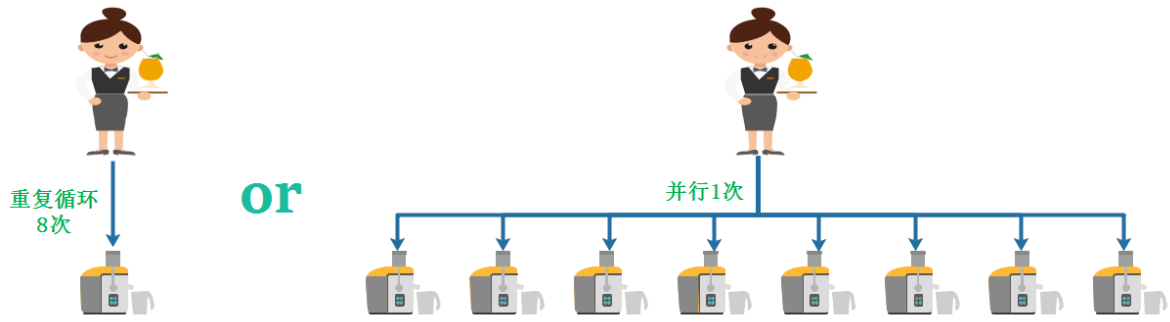
在计算机系统的体系结构中，存储系统是一种层次结构，典型服务器计算机的存储层次结构如上图，上图表述了CPU、CPU三级缓存、内存、磁盘数据容量与数据读取速度对比，我们可以看出存储媒介距离CPU越近，则访问数据的速度越快。

注意：缓存就是数据交换的缓冲区，缓存往往都是RAM(断电即掉的非永久储存),它们的作用就是帮助硬件更快地响应。CPU缓存的定义为CPU与内存之间的临时数据交换器，它的出现是为了解决CPU运行处理速度与内存读写速度不匹配的矛盾，CPU缓存一般直接跟CPU芯片集成或位于主板总线互连的独立芯片上，现阶段的CPU缓存一般直接集成在CPU上。CPU往往需要重复处理相同的数据、重复执行相同的指令，如果这部分数据、指令，CPU能在CPU缓存中找到，CPU就不需要从内存或硬盘中再读取数据、指令，从而减少了整机的响应时间。

由上图可知，从内存读取数据速度比磁盘读取数据速度要快1000倍，从CPU缓存中读取数据的速度比从内存中读取数据的速度最快要快100倍，从CPU寄存器中读取数据的速度为300ps(1000ps 皮秒 = 1ns)，比CPU缓存要快3倍还多。从寄存器中访问数据的速度，是从内存访问数据速度的300倍，是从磁盘中访问数据速度的30万倍。

如果能从CPU寄存器中访问数据对程序的性能提升意义非凡，向量化执行就是在寄存器层面操作数据，为上层应用程序的性能带来了指数级的提升。

何为向量化执行？向量化执行，可以简单地看作一项消除程序中循环的优化。 这里用一个形象的例子比喻。小胡经营了一家果汁店，虽然店里的鲜榨苹果汁深受大家喜爱，但客户总是抱怨制作果汁的速度太慢。小胡的店里只有一台榨汁机，每次他都会从篮子里拿出一个苹果，放到榨汁机内等待出汁。如果有8个客户，每个客户都点了一杯苹果汁，那么小胡需要重复循环8次上述的榨汁流程，才能榨出8杯苹果汁。如果制作一杯果汁需要5分钟，那么全部制作完毕则需要40分钟。为了提升果汁的制作速度，小胡想出了一个办法。他将榨汁机的数量从1台增加到了8台，这么一来，他就可以从篮子里一次性拿出8个苹果，分别放入8台榨汁机同时榨汁。此时，小胡只需要5分钟就能够制作出8杯苹果汁。**为了制作n杯果汁，非向量化执行的方式是用1台榨汁机重复循环制作n次，而向量化执行的方式是用n台榨汁机只执行1次。**



为了实现向量化执行，需要利用CPU的SIMD指令，SIMD的全称是Single Instruction Multiple Data，即用**单条指令操作多条数据**。现代计算机系统概念中，它是通过数据并行以提高性能的一种实现方式(其他的还有指令级并行和线程级并行)，它的**原理是在CPU寄存器层面实现数据的并行操作。**

ClickHouse列式存储除了降低IO和存储的压力之外，还为向量化执行做好了铺垫，除了读取磁盘速度快之外，ClickHouse还利用SSE4.2指令集实现向量化执行，为处理数据提升很大效率。

1.1.3.5 关系模型与标准SQL查询

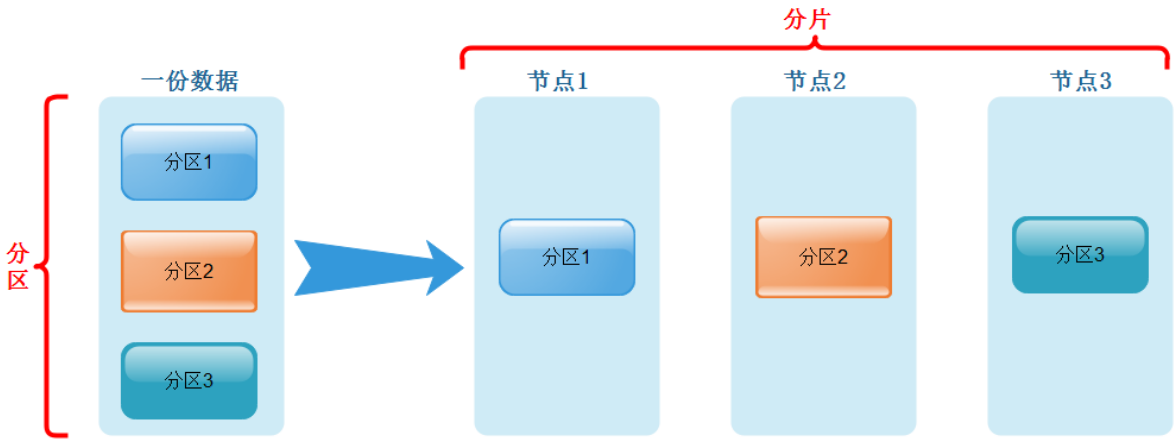
相比HBase和Redis这类NoSQL数据库，ClickHouse使用关系模型描述数据并提供了传统数据库的概念(数据库、表、视图和函数等)。ClickHouse完全使用SQL作为查询语言(支持GROUP BY、ORDER BY、JOIN、IN等大部分标准SQL)，ClickHouse提供了标准协议的SQL查询接口，可以与第三方分析可视化系统无缝集成对接。在SQL解析方面，ClickHouse是大小写敏感，SELECT a 和 SELECT A所代表的语义不同。

1.1.3.6 多样化的表引擎

与MySQL类似，ClickHouse也将存储部分进行了抽象，把存储引擎作为一层独立的接口。ClickHouse拥有各种表引擎，每种表引擎决定不同的数据存储方式。其中每一种表引擎都有着各自的特点，用户可以根据实际业务场景的要求，选择合适的表引擎使用。将表引擎独立设计的好处是通过特定的表引擎支撑特定的场景，十分灵活，对于简单的场景，可直接使用简单的引擎降低成本，而复杂的场景也有合适的选择。

1.1.3.7 多线程与分布式

向量化执行是通过数据级并行的方式提升了性能，多线程处理是通过线程级并行的方式实现了性能的提升。相比基于底层硬件实现的向量化执行SIMD，线程级并行通常由更高层次的软件层面控制，目前市面上的服务器都支持多核心多线程处理能力。由于SIMD不适合用于带有较多分支判断的场景，ClickHouse也大量使用了多线程技术以实现提速，以此和向量化执行形成互补。ClickHouse在数据存取方面，既支持分区(纵向扩展，利用多线程原理)，也支持分片(横向扩展，利用分布式原理)，可以说是将多线程和分布式的技术应用到了极致。



1.1.3.8 多主架构

HDFS、Spark、HBase和Elasticsearch这类分布式系统，都采用了Master-Slave主从架构，由一个管控节点作为Leader统筹全局。而ClickHouse则采用Multi-Master多主架构，集群中的每个节点角色对等，客户端访问任意一个节点都能得到相同的效果。这种多主的架构有许多优势，例如对等的角色使系统架构变得更加简单，不用再区分主控节点、数据节点和计算节点，集群中的所有节点功能相同。所以它天然规避了单点故障的问题，非常适合用于多数据中心、异地多活的场景。

1.1.3.9 交互式查询

Hive,SparkSQL,HBase等都支持海量数据的查询场景，都拥有分布式架构，都支持列存、数据分片、计算下推等特性。ClickHouse在设计上吸取了以上技术的优势，例如：Hive、SparkSQL无法保证90%以上的查询在秒级内响应，在大数据量复杂查询下需要至少分钟级的响应时间，而HBase可以对海量数据做到交互式查询，由于不支持标准SQL在对数据做OLAP聚合分析时显得捉襟见肘。ClickHouse吸取以上各个技术的优势，在复杂查询的场景下，它也能够做到极快响应，且无须对数据进行任何预处理加工。

1.1.3.10 数据分片与分布式查询

数据分片是将数据进行横向切分，这是一种在面对海量数据的场景下，解决存储和查询瓶颈的有效手段，是一种分治思想的体现。ClickHouse支持分片，而分片则依赖集群。每个集群由1到多个分片组成，而每个分片则对应了ClickHouse的1个服务节点。分片的数量上限取决于节点数量（1个分片只能对应1个服务节点）。

ClickHouse拥有高度自动化的分片功能。ClickHouse提供了本地表 （Local Table）与分布式表（Distributed Table）的概念。一张本地表等同于一份数据的分片。而分布式表本身不存储任何数据，它是本地表的访问代理，其作用类似分库中间件。借助分布式表，能够代理访问多个数据分片，从而实现分布式查询。

这种设计类似数据库的分库和分表，十分灵活。例如在业务系统上线的初期，数据体量并不高，此时数据表并不需要多个分片。所以使用单个节点的本地表（单个数据分片）即可满足业务需求，待到业务增长、数据量增大的时候，再通过新增数据分片的方式分流数据，并通过分布式表实现分布式查询。

1.2 ClickHouse安装

Clickhouse官网为：[https://clickhouse.com/],在官网中可以看到ClickHouse可以基于多种方式安装，rpm安装、tgz安装包安装、docker镜像安装、源码编译安装等。这里我们使用rpm安装包安装。(https://clickhouse.com/，在官网中可以看到ClickHouse可以基于多种方式安装，rpm安装、tgz安装包安装、docker镜像安装、源码编译安装等。这里我们使用rpm安装包安装。)目前Clickhouse仅支持Linux系统且cpu必须支持SSE4.2指令集，可以通过以下命令查询Linux是否支持：

```
grep -q sse4_2 /proc/cpuinfo && echo "SSE 4.2 supported" || echo "SSE 4.2 not
supported"
```

如果服务器不支持SSE4.2指令集，则不能下载预编译安装包，需要通过源码编译特定版本进行安装。

1.2.1 rpm安装包下载

ClickHouse rpm安装包查询地址为:https://packagecloud.io/Altinity/clickhouse，这里需要在linux中使用wget命令下载对应的clickHouse版本。选择一台服务器创建/software目录并进入此目录，在当前目录下执行如下命令下载ClickHouse需要的rpm安装包，这里只需要下载以下四个rpm安装包即可。

```
wget --content-disposition
https://packagecloud.io/Altinity/clickhouse/packages/el/7/clickhouse-common-
static-20.8.3.18-1.el7.x86_64.rpm/download.rpm
wget --content-disposition
https://packagecloud.io/Altinity/clickhouse/packages/el/7/clickhouse-server-
common-20.8.3.18-1.el7.x86_64.rpm/download.rpm
wget --content-disposition
https://packagecloud.io/Altinity/clickhouse/packages/el/7/clickhouse-server-
20.8.3.18-1.el7.x86_64.rpm/download.rpm
wget --content-disposition
https://packagecloud.io/Altinity/clickhouse/packages/el/7/clickhouse-client-
20.8.3.18-1.el7.x86_64.rpm/download.rpm
```

我们也可以从官网给定的下载rpm包的地址手动下载最新的clickhouse rpm安装包，地址如下：https://repo.yandex.ru/clickhouse/rpm/stable/x86_64/

注意：这里从clickhouse19.4版本之后，只需要下载3个rpm安装包即可，分别如下：

```
clickhouse-common-static-21.9.4.35-2.x86_64.rpm
clickhouse-server-21.9.4.35-2.noarch.rpm
clickhouse-client-21.9.4.35-2.noarch.rpm
```

1.2.2 单节点安装

选择一台服务器，将下载好的clickHouse安装包直接安装即可，这里选择clickhouse21.9.4版本进行安装，安装顺序如下：

```
rpm -ivh clickhouse-common-static-21.9.4.35-2.x86_64.rpm
rpm -ivh clickhouse-server-21.9.4.35-2.noarch.rpm
rpm -ivh clickhouse-client-21.9.4.35-2.noarch.rpm
这里也可以在当前节点直接执行如下命令，也可以按照依赖关系安装各个rpm包：
rpm -ivh ./clickhouse-*.rpm
... ...
Create user clickhouse.clickhouse with datadir /var/lib/clickhouse
```

1.2.2.1 目录介绍

安装完成之后会生成如下对应的目录，每个目录的介绍如下：

- /etc/clickhouse-server : 服务端的配置文件目录，包括全局配置config.xml 和用户配置users.xml。
- /var/lib/clickhouse : 默认的数据存储目录，通常会修改，将数据保存到大容量磁盘路径中。
- /var/log/cilckhouse-server : 默认保存日志的目录，通常会修改，将数据保存到大容量磁盘路径中。
- 在/usr/bin下会有可执行文件：
  - clickhouse:主程序可执行文件
  - clickhouse-server:一个指向clickhouse可执行文件的软连接，供服务端启动使用。
  - clickhouse-client:一个指向clickhouse可执行文件的软连接，供客户端启动使用。

1.2.2.2 启动&停止服务

启动clickhouse-server服务：

```
service clickhouse-server start
```

启动clickhouse服务后可以使用命令行客户端连接到服务：

```
#client客户端连接到ch服务
clickhouse-client
或者使用命令：
clickhouse-client --host localhost --port 9000
ClickHouse client version 20.8.3.18.
Connecting to localhost:9000 as user default.
Connected to ClickHouse server version 20.8.3 revision 54438.
node1 :)

#查看9000 端口占用情况
[root@node5 bin]# yum install net-tools
[root@node5 bin]# netstat -tunlp |grep 9000

#查看当前所有数据库
show databases;
┌─name─────────────────────────────┐
| _temporary_and_external_tables |
| default                        |
| system                         |
```

```
└──────────────────────────────┘
#查看当前使用的数据库
select database();
┌─database()─┐
| default     |
└────────────┘
#退出客户端
quit;
```

关闭ClickHouse服务：

```
service clickhouse-server stop
```

1.2.3 clickhouse分布式集群安装

clickhouse分布式集群安装选择三台节点，分别为node1,node2,node3，详细安装步骤如下：

1. **选择三台clickhouse节点，在每台节点上安装clickhouse需要的安装包**

这里选择node1、node2,node3三台节点，上传安装包，分别在每台节点上执行如下命令安装clickhouse:

```
rpm -ivh ./clickhosue-common-static-21.9.4.35-2.x86_64.rpm
#注意在安装以下rpm包时，让输入密码，可以直接回车跳过
rpm -ivh ./clickhouse-server-21.9.4.35-2.noarch.rpm
rpm -ivh clickhouse-client-21.9.4.35-2.noarch.rpm
```

1. **安装zookeeper集群并启动。**

搭建clickhouse集群时，需要使用Zookeeper去实现集群副本之间的同步，所以这里需要zookeeper集群，zookeeper集群安装后可忽略此步骤。

1. **配置外网可访问**

在每台clickhouse节点中配置/etc/clickhouse-server/config.xml文件第164行<listen_host>，如下：

```
<listen_host>::1</listen_host>
#注意每台节点监听的host名称配置当前节点host
<listen_host>node1</listen_host>
```

1. **在每台节点创建metrika.xml文件，写入以下内容**

在node1、node2、node3节点上/etc/clickhouse-server/config.d路径下下配置metrika.xml文件，默认clickhouse会在/etc路径下查找metrika.xml文件，但是必须要求metrika.xml上级目录拥有者权限为clickhouse ，所以这里我们将metrika.xml创建在/etc/clickhouse-server/config.d路径下，config.d目录的拥有者权限为clickhouse。

在metrika.xml中我们配置后期使用的clickhouse集群中创建分布式表时使用3个分片，每个分片有1个副本，配置如下：

vim /etc/clickhouse-server/config.d/metrika.xml:

```
<yandex>
    <remote_servers>
        <clickhouse_cluster_3shards_1replicas>
            <shard>
                <internal_replication>true</internal_replication>
```

```
                <replica>
                    <host>node1</host>
                    <port>9000</port>
                </replica>
            </shard>
            <shard>
                <internal_replication>true</internal_replication>
                <replica>
                    <host>node2</host>
                    <port>9000</port>
                </replica>
            </shard>
            <shard>
                <internal_replication>true</internal_replication>
                <replica>
                    <host>node3</host>
                    <port>9000</port>
                </replica>
            </shard>
        </clickhouse_cluster_3shards_1replicas>
    </remote_servers>

    <zookeeper>
        <node index="1">
            <host>node3</host>
            <port>2181</port>
        </node>
        <node index="2">
            <host>node4</host>
            <port>2181</port>
        </node>
        <node index="3">
            <host>node5</host>
            <port>2181</port>
        </node>
    </zookeeper>
    <macros>
        <shard>01</shard>
        <replica>node1</replica>
    </macros>
    <networks>
        <ip>::/0</ip>
    </networks>
    <clickhouse_compression>
        <case>
            <min_part_size>10000000000</min_part_size>
            <min_part_size_ratio>0.01</min_part_size_ratio>
            <method>lz4</method>
        </case>
    </clickhouse_compression>
</yandex>
```

对以上配置文件中配置项的解释如下:

- remote_servers:

clickhouse集群配置标签，固定写法。**注意：这里与之前版本不同，之前要求必须以clickhouse开头，新版本不再需要。**

- clickhouse_cluster_3shards_1replicas:

配置clickhouse的集群名称，可自由定义名称，注意集群名称中不能包含点号。这里代表集群中有3个分片，每个分片有1个副本。

分片是指包含部分数据的服务器，要读取所有的数据，必须访问所有的分片。

副本是指存储分片备份数据的服务器，要读取所有的数据，访问任意副本上的数据即可。

- shard：

分片，一个clickhouse集群可以分多个分片，每个分片可以存储数据，这里 **分片可以理解为clickhouse机器中的每个节点，1个分片只能对应1服务节点** 。这里可以配置一个或者任意多个分片，在每个分片中可以配置一个或任意多个副本，不同分片可配置不同数量的副本。如果只是配置一个分片，这种情况下查询操作应该称为远程查询，而不是分布式查询。

- replica：

每个分片的副本，默认每个分片配置了一个副本。也可以配置多个，副本的数量上限是由clickhouse节点的数量决定的。如果配置了副本，读取操作可以从每个分片里选择一个可用的副本。如果副本不可用，会依次选择下个副本进行连接。该机制利于系统的可用性。

- internal_replication：

默认为false,写数据操作会将数据写入所有的副本，设置为true,写操作只会选择一个正常的副本写入数据，数据的同步在后台自动进行。

- zookeeper：

配置的zookeeper集群，**注意：与之前版本不同，之前版本是“zookeeper-servers”。**

- macros：

区分每台clickhouse节点的宏配置，macros中标签代表当前节点的分片号，标签代表当前节点的副本号，这两个名称可以随意取，后期在创建副本表时可以动态读取这两个宏变量。注意：每台clickhouse节点需要配置不同名称。

- networks：

这里配置ip为“::/0”代表任意IP可以访问，包含IPv4和IPv6。

注意：允许外网访问还需配置/etc/clickhouse-server/config.xml 参照第三步骤。

- clickhouse_compression：

MergeTree引擎表的数据压缩设置，min_part_size：代表数据部分最小大小。min_part_size_ratio：数据部分大小与表大小的比率。method：数据压缩格式。

注意：需要在每台clickhouse节点上配置metrika.xml文件，并且修改每个节点的 macros配置名称。

```
#node2节点修改metrika.xml中的宏变量如下：
    <macros>
        <shard>02</replica>
        <replica>node2</replica>
    </macros>

#node3节点修改metrika.xml中的宏变量如下：
<macros>
        <shard>03</replica>
        <replica>node3</replica>
    </macros>
```

1. **在每台节点上启动/查看/重启/停止clickhouse服务**

首先启动zookeeper集群，然后分别在node1、node2、node3节点上启动clickhouse服务，这里每台节点和单节点启动一样。启动之后，clickhouse集群配置完成。

```
#每台节点启动Clickchouse服务
service clickhouse-server start

#每台节点查看clickhouse服务状态
service clickhouse-server status

#每台节点重启clickhouse服务
service clickhouse-server restart

#每台节点关闭Clikchouse服务
service clickhouse-server stop
```

1. **检查集群配置是否完成**

在node1、node2、node3任意一台节点进入clickhouse客户端，查询集群配置：

```
#选择三台clickhouse任意一台节点，进入客户端
clickhouse-client
#查询集群信息，看到下图所示即代表集群配置成功。
node1 :) select * from system.clusters;
```

```
┌─cluster─────────────────────────┬─shard_num─┬─shard_weight─┬─replica_num─┬─host_name─┬─host_address──┬─port─┬─is_local─┬─user────┬─default_database─┬─errors_count─┬─slowdowns_count─┬─estimated_recovery_time─┐
│ clickhouse_cluster_3shards_1replicas │         1 │            1 │           1 │ node1     │ 192.168.179.4 │ 9000 │        1 │ default │                 │            0 │               0 │                       0 │
│ clickhouse_cluster_3shards_1replicas │         2 │            1 │           1 │ node2     │ 192.168.179.5 │ 9000 │        0 │ default │                 │            0 │               0 │                       0 │
│ clickhouse_cluster_3shards_1replicas │         3 │            1 │           1 │ node3     │ 192.168.179.6 │ 9000 │        0 │ default │                 │            0 │               0 │                       0 │
└─────────────────────────────────┴───────────┴──────────────┴─────────────┴───────────┴───────────────┴──────┴──────────┴─────────┴─────────────────┴──────────────┴─────────────────┴─────────────────────────┘
```

```
#查询集群信息，也可以使用如下命令
node1 :) select cluster,host_name from system.clusters;
```

```
┌─cluster─────────────────────────────────────┬─host_name─┐
│ clickhouse_cluster_3shards_1replicas        │ node1     │
│ clickhouse_cluster_3shards_1replicas        │ node2     │
│ clickhouse_cluster_3shards_1replicas        │ node3     │
│ test_cluster_two_shards                     │ 127.0.0.1 │
│ test_cluster_two_shards                     │ 127.0.0.2 │
│ test_cluster_two_shards_internal_replication │ 127.0.0.1 │
│ test_cluster_two_shards_internal_replication │ 127.0.0.2 │
└─────────────────────────────────────────────┴───────────┘
```

1.3 客户端命令行参数

我们可以通过clickhouse client来连接启动的clickhouse服务，连接服务时，我们可以指定以下参数，这里指定的参数会覆盖默认值和配置文件中的配置。

参数	解释
--host, -h	服务端的host名称, 默认是localhost。您可以选择使用host名称或者IPv4或IPv6地址。
--port	连接的端口，默认值：9000。注意HTTP接口以及TCP原生接口使用的是不同端口。
--user, -u	用户名。默认值：default。
--password	密码。默认值：空字符串。
--query，-q	使用非交互模式查询。
--database, -d	默认当前操作的数据库. 默认值：服务端默认的配置（默认是default）。
--multiline, -m	如果指定，允许多行语句查询（Enter仅代表换行，不代表查询语句完结）。
--time, -t	如果指定，**非交互模式下**会打印查询执行的时间到stderr中。
--stacktrace	如果指定，如果出现异常，会打印堆栈跟踪信息。
--config-file	配置文件的名称。
--multiquery，-n	使用非交互模式查询数据时，可以分号隔开多个sql语句。

Ø **--host，-h:**

使用-h指定ip或者host名称时，需要在/etc/clickhouse-server/config.xml配置文件中114行配置：<listen_host>::</listen_host> ，代表可以任意ip可访问。配置完成后需要重启当期clickhouse节点生效。

```
clickhouse-client  -h node1
ClickHouse client version 20.8.3.18.
Connecting to node1:9000 as user default.
Connected to ClickHouse server version 20.8.3 revision 54438.
```

Ø **--query，-q**

```
clickhouse-client -q "show databases"
_temporary_and_external_tables
default
system
```

Ø **--database, -d:**

```
clickhouse-client -d "system" -q "show tables"
aggregate_function_combinators
asynchronous_metric_log
asynchronous_metrics
build_options
... ....
```

Ø **--multiline, -m:**

```
clickhouse-client -m

node1 :) select
:-] 1+1
:-] ;

SELECT 1 + 1
┌─plus(1, 1)─┐
│          2 │
└────────────┘
1 rows in set. Elapsed: 0.004 sec.
```

Ø **--time, -t:**

```
clickhouse-client -t -q "show databases"
_temporary_and_external_tables
default
system
0.004
```

Ø **--stacktrace:**

```
clickhouse-client --stacktrace
ClickHouse client version 20.8.3.18.
Connecting to localhost:9000 as user default.
Connected to ClickHouse server version 20.8.3 revision 54438.

node1 :) use aaa;
USE aaa
Received exception from server (version 20.8.3):
Code: 81. DB::Exception: Received from localhost:9000. DB::Exception: Database
aaa doesn't exist. Stack trace:
0.Poco::Exception::Exception(std::__1 ... ....
... ....
```

Ø **--multiquery，-n**

```
[root@node1 ~]# clickhouse-client  -n -q "show databases;use default;"
_temporary_and_external_tables
default
system
```

1.4 数据类型

ClickHouse提供了许多数据类型，它们可以划分为基础类型、复合类型和特殊类型。我们可以在system.data_type_families表中检查数据类型名称以及是否区分大小写。这个表中存储了ClickHouse支持的所有数据类型。

```
select * from system.data_type_families limit 10;
SELECT *
FROM system.data_type_families
LIMIT 10

┌─name──────────┬─case_insensitive─┬─alias_to─┐
| Polygon       |                0 |          |
| Ring          |                0 |          |
| MultiPolygon  |                0 |          |
| IPv6          |                0 |          |
| IntervalSecond|                0 |          |
| IPv4          |                0 |          |
| UInt32        |                0 |          |
| IntervalYear  |                0 |          |
| IntervalQuarter|               0 |          |
| IntervalMonth |                0 |          |
└───────────────┴──────────────────┴──────────┘

10 rows in set. Elapsed: 0.004 sec.
```

下面介绍下常用的数据类型，ClickHouse与Mysql、Hive中常用数据类型的对比图如下：

MySQL	**Hive**	**ClickHouse(区分大小写)**
byte	TINYINT	Int8
short	SMALLINT	Int16
int	INT	Int32
long	BIGINT	Int64
varchar	STRING	String
timestamp	TIMESTAMP	DateTime
float	FLOAT	Float32
double	DOUBLE	Float64
boolean	BOOLEAN	无