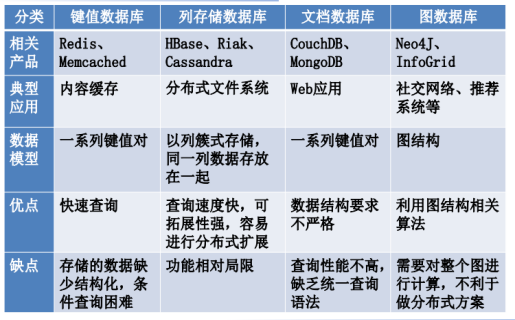
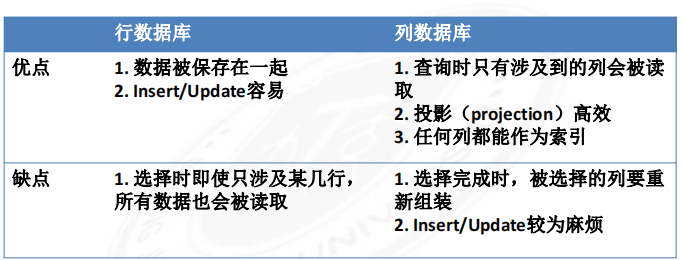
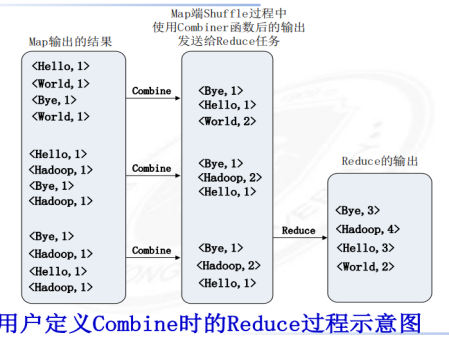
**什么是数据？数据是所有能输入到计算机并被计算机程序处理的符号的总称。人们通过观察现实世界中的自然现象、人类活动，都可以形成数据。/分类：结构化，半~，非~./作用：解释现象，发现规律，进行预测。/数据到信息到知识到智慧。/大数据：大数据指的是那些增长很快，以至于难以用传统（数据库）软件工具捕获、存储、管理、共享、分析和可视化的数据集/4v,5v:规模巨大体量巨大（Volume）、种类繁多（Variety）、增长速度快（Velocity）和变化频繁（Variability）+value（数据价值）/思维方式的改变：采样到全样, 精确非精确,因果关联,个体群体,随机规律/作用：经济社会科学（三种模式:实验，理论分析，计算；第四模式:Data Intensive Science Discovery/处理过程：数据获取:需要对数据进行变换、清洗等预处理，输出满足数据应用要求的数据；数据管理:对数据进行分类、编码、存储索引和查询；数据分析:描述性分析、诊断性分析、预测生分析和规范性分析数据可视化与交互分析:帮助业务人员而非数据处理专家更好的理解数据分析的结果/计算：规模化，蛮算变巧算，3I计算特征：近似（数据异构多源，多噪声；目标:寻找数据关联趋势宏观特征；·传统精确处理思路不适用·允许解的一定范围区间近似），归纳（数据分布在多源数据集中，多源数据间隐含关联特性；单源数据不足需修正处理偏差·从传统还原法到归纳法），增量（数据持续产生，且不断更新，数据处理实时性要求高；·难以形成大数据统一视图·全量式批处理难以满足要求；）**

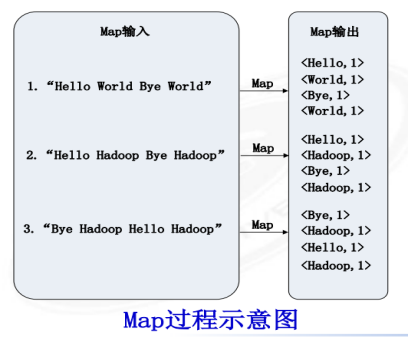
**2:大数据技术支撑：/来源：18月数据翻一番；主体划分有：-少量企业应用产生的数据如关系型数据库中的数据和数据仓库中的数据等。-大量人产生的数据如推特、微博、通信软件、移动通信数据、电子商务在线交易日志数据、企业应用的相关评论数据等。巨量机器产生的数据如应用服务器日志、各类传感器数据、图像和视频监控数据、扫描数据等，来源划分：互联网公司，各电信石化系统，医疗交通领域，制造业等传统行业；数据存储形式：• 结构化数据简单来说就是数据库，如企业ERP、财务系统、医疗HIS数据库、教育一卡通、政府行政审批、其他核心数据库等 非结构化数据包括所有格式的办公文档、文本、图片、XML、HTML、各类报表、图像和音频、视频信息/特征：3S：Size:数据的大小Speed:数据的处理速度，数据的结构化Structure/表现形态：多源性:大数据来源的复杂性。网络技术的迅猛发展使得数据产生的途径多样化。大数据结构的复杂性。非结构化数据的格式多样化，而这些非结构化数据中可能蕴藏着非常有价值的信息。实时性:大数据的实时性，体现在数据更新的实时性。如何及时有效、全面的捕获到互联网、物联网、云计算上产生的大量的不同来源的数据是会直接影响数据价值体现的关键因素。不确定性:体现的是数据的不确定性。原始数据的不准确以及数据采集处理粒度、应用需求与数据集成和展示等因素使得数据在不同尺度、不同维度上都有不同程度的不确定性/应用：零售：消费者需求，偏好趋势；金融：商业价值，保险精算水平，交易习惯和行为分析；医疗：提出医疗方案，疾病特点数据库建立；教育：个体化学习；农业；环境；智慧城市（大数据技术可以了解经济发展情况、各产业发展情况、消费支出和产品销售情况等，依据分析结果，科学地制定宏观政策，平衡各产业发展，避免产能过剩，有效利用自然资源和社会资源，提高社会生产效率。大数据技术也能帮助政府进行支出管理，透明合理的财政支出将有利于提高公信力和监督财政支出）**

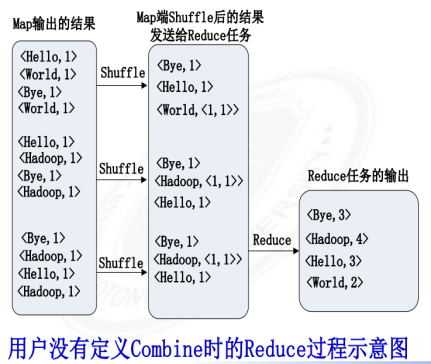
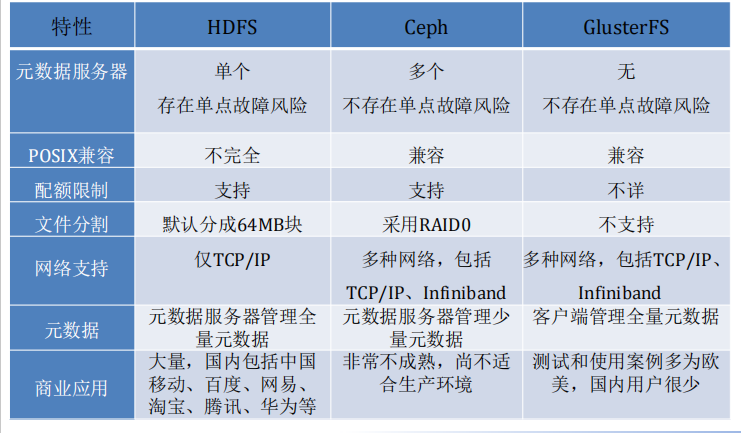
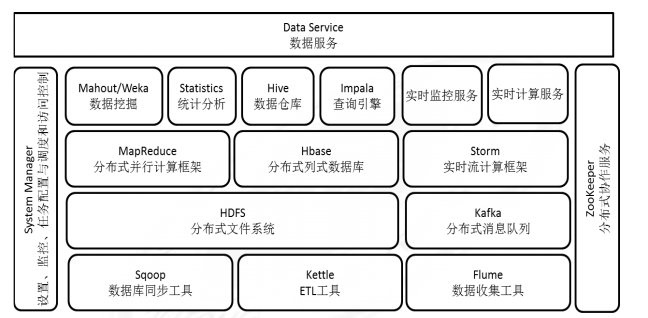
**3:挑战：从4v入手，传统：产生（被动采集和主动生成），采集：采样密度较低，采样数据有限；可对需要分析事件的利用大数据平台，数据进行密度采样，精确获取事件全局数据，数据源：数据源获取较为孤立，不司数据之间添加的数据整合难度较大；利用大数据技术，通过分布式技术、布式文件系统、分布式数据库等技术对多个数据源获取的数据进行整合处理，处理方式：大多采用离线处理方式，对生成的数据集中分析处理；较大的数据源、响应时间要求低的应用可以采取批处理方式集中计算;响应时间要求高的实时数据处理采用流处理的方式进行实时计算，并通过对历史数据的分析进行预测分析/内部数据和互联网/优势：构建数据驱动应用，推进拓展价值实现；统一数据规范标准，推动数据共享开放；重视数据安全管理，完善数据安全保障；推进数据融合管理，增加数据语义厚度/对数据进行有效的整合将成为是否能够对内部数据进行有效利用和关键，ETL（是数据的抽取、转换、装载的过程，负责完成数据从数据源向目标数据仓库的转化。即用户从数据源抽取所需的数据，经过数据清洗，按照预先定义的数据仓库模型，最终将数据加载入数据仓库)重要处理手段/抽取:增量捕捉变化数据：日志对比时间戳触发器全变对比/etl引擎或数据库转换，规则：字段级转化，清洁净化，多数据源整合，聚合汇总/装载:sql,批量装载/数据清洗:发现不准确、不完整或不合理数据，修补或移除以提高数据质量:定义错误类型搜索并标识错误实例改正错误文档记录错误实例和错误类型修改数据录入程序以减少未来的错误/互联网d特征：多源异构性交互性时效性社会性突发性高噪音/爬虫：一种自动搜集互联网信息的程序，它为搜索引擎从万维网上下载网页，是搜索引擎的重要组成;控制器解析器资源库,步骤:采集分析过滤/抓取策略：深度(从URL池中选择某URL，然后按深度优先遍历以该URL为根节点的所有URL网页内容，然后取出URL池中下一个URL，继续上述策略循环至URL池遍历完优点是设计简易，抓取深度大缺点是容易导致无限抓取，使得爬取过程无法收敛)广度(按照广度优先搜索思想，逐层抓取URL池中的每一个URL内容并将每一层的扇出URL纳入URL池中，按照广度优先策略循环遍历。优点是抓取宽度广，抓取过程容易控制，有效减轻服务器的负载缺点是容易造成URL大量聚集而导致URL池溢出)partialpagerank(Partial PageRank策略借鉴了PageRank算法的思想:对于已经下载的网页，连同待抓取URL队列中的URL，形成网页集合，计算每个页面的PageRank值，计算完之后，将待抓取URL队列中的URL按照PageRank值的大小排列:并按照该顺序抓取页面。优点是抓取优先级可控缺点是由于广告和作弊链接的存在，容易导致PageRank值不能完全刻画其重要程度，从而导致实际抓取数据无效)OPIC(该算法实际上也是对页面进行一个重要性打分。在算法开始前，给所有页面一个相同的初始现金(cash)。当下载了某个页面P之后，将P的现金分摊给所有从P中分析出的链接，并且将P的现金清空。对于待抓取URL队列中的所有页面按照现金数进行排序。优点是计算速度快于PartialPageRank策略，适合实时计算场景缺点是受初始值影响大)**

**4:LFS（本地文件系统）,DFS:存储方式:直接,分块/存储结构:树，master-slaver主从,数据检索备：慢快；数据备份：否，是；数据安全：低高；数据对象：不适用于大规模文件；存储空间：不需要额外的存储空间；访问方式：直接访问；加载后访问,复杂性：低高/分布式改变数据的存储管理方式,优异的数据备份安,规模可扩展/分类：面向以大文件、块数据顺序读写为特点的数据分析业务，其典型代表是Apache旗下的HDFS；主要服务于通用文件系统需求并支持标准的可移植操作系统接口(Portable OperatingSystem Interface ofUNIX，POSIX)，其代表包括Ceph和GlusterFS，两种分类只是专注点不同不是唯一用途/hadoop特点：存储和高速运算；核心设计:HDFS和MapReduce，HDFS(HadoopDistributed File System)为海量数据提供存储,MapReduce为海量数据提供计算/功能：为数据的存储、管理和出错处理。设计的目的是用于可靠地存储大规模的数据集，并提高用户访问数据的效率/4点：适合大文件存储和处理：可处理的文件规模可达到百MB乃至数百TB，目前应用已到PB级；集群规模可动态扩展：存储节点可在运行状态下加入到集群中，集群仍然可以正常地工作；能有效保证数据一致性：基于“一次写入，多次读取”设计，简化处理文件访问方式，当一个文件创建、写入并关闭后就不能再修改；数据的吞吐量大跨平台移植性好：采用数据流式读写的方式，用以增加数据的吞吐量。具有很好的跨平台移植性，源代码开放/名节点：名称节点，用于维护和记录文件系统的命名空间，保存了两个核心的数据结构，即FsImage和EditLog。FsImmage用于维护文件系统树以及文件树中所有的文件和文件夹的元数据。EditLog:记录了所有针对文件的创建、删除、重命名等操作/数据节点：分布式文件系统HDFS的工作节点，负责数据的存储和读取，会根据客户端或者是名节点的调度来进行数据的存储和检索，并且向名节点定期发送自己所存储的块的列表。响应文件系统客户端发出的读写请求，同时在名节点的指导下执行数据库的创建、删除及复制。每个数据节点中的数据会被保存在各自节点的本地Linux文件系统中/HDFS客户端:是用户操作HDFS最常用的方式，HDFS在部署时都提供了客户端。是一个库，暴露了HDFS文件系统接口。客户端通过一个可配置的端口向名节点主动发起TCP连接，并使用客户端协议与名节点进行交互。此外，HDFS也提供了Java API，作为应用程序访问文件系统的客户端编程接口/命名空间管理:包含目录、文件和块。在HDFS1.0体系结构中，在整个HDFS集群中只有一个命名空间，并且只有唯一一个名节点，该节点负责对这个命名空间进行管理.HDFS使用的是传统的分级文件体系，因此，用户可以像使用普通文件系统一样，创建、删除目录和文件，在目录间转移文件，重命名文件等/HDFS通信协议:HDFS是一个部署在集群上的分布式文件系统，因此很多数据需要通过网络进行传输。所有xx协议都是构建在TCP/IP协议基础之上；客户端通过一个可配置的端口向名节点主动发起TCP连接，并使用客户端协议与名节点进行交互。名节点和数据节点之间则使用数据节点协议进行交瓦。客户端与数据节点的交互则是通过RPC(RemoteProcedure Call)来实现/概述：HDFS采用了主从(Master/Slave)架构，HDFS集群包括一个名节点和若干个数据节点。名节点作为中心服务器，负责管理文件系统的命名空间及客户端对文件的访问。集群中的数据节点负责处理文件系统客户端的读/写请求，在名节点的统一调度下进行数据块的创建、删除和复制等操作。所有的HDFS通信协议都是构建在TCP/IP协议基础之上。每个数据节点的数据实际上是保存在本地Linux文件系统中。有个后背主服务器定期备份，对其他节点不可见，故障使用/设计原理：元数据与数据分离-名节点与数据节点分离。主从结构-主控件控制从控件，一次写入多次读取HDFS中的文件只能写入一次，确保了数据的一致性。移动计算比移动数据更划算将运算的执行移到离它要处理的数据更近的地方/冗余数据：为了保证系统的容错性和可用性，HDFS采用了多副本方式对数据进行冗余存储，通常一个数据块的多个副本会被分布到不同的数据节点上。多副本优点：1)加快数据传输速度2)容易检查数据错误3)保证数据可靠性/存取策略:1)存放第一个副本:放置在上传文件的数据节点;如果是集群，提交，则随机挑选一台磁盘不太满、CPU不太忙的节点。第二个副本:放置在与第一个副本不同的机架的节点上。第三个副本:与第一个副本相同机架的其他节点上。更多副本:随机节点2）读取：HDFS提供了API接口以确定数据节点所属的机架ID，客户端也可以调用API获取自己所属的机架ID，当客户端读取数据时，从名节点获得数据块不同副本的存放位置列表，列表中包含了副本所在的数据节点，可以调用API来确定客户端和这些数据节点所属的机架ID当发现某个数据块副本对应的机架ID和客户端对应的机架ID相同时，就优先选择该副本读取数据，如果没有发现，就随机选择一个副本读取数据。/恢复：1：名节点出错名节点保存了所有的元数据信息，其中，最核心的两大数据结构是FsImage和Editlog，若这两个文件发生损坏，那么整个HDFS实例将失效。将核心文件同步复制到备份服务器SecondaryNameNode上当名节点出错时，可根据备份服务器中的FsImage和Editlog数据进行恢复/2:数据节点出错每个数据节点会定期向名节点发送“心跳”信息，向名节点报告自己的状态。当数据节点发生故障，或者网络发生断网时，名节点就无法收到来自数据节点的心跳信息，这时，这些数据节点就会被标记为“宕机”，节点上面的所有数据都会被标记为“不可读”，名节点不会再给它们发送任何I/0请求。名节点会定期检查，一旦发现某个数据块的副本数量小于冗余因子，就会启动数据冗余复制，为它生成新的副/3数据出错：网络传输和磁盘错误。在文件被创建时，客户端就会对每一个文件块进行信息摘录，并把这些信息写入到同一个路径的隐藏文件里面客户端读取文件时，会先读取信息文件，然后，利用该信息文件对每个读取的数据块进行校验，如果校验出错客户端就会请求到另外一个数据节点读取该文件块，并且向名节点报告这个文件块有错误，名节点会定期检查并且重新复制这个块/存储原理：名节点与数据节点-从HDFS系统的内部架构来看，一个文件被分成多个文件块储存在数据节点集上。文件系统命名空间：HDFS支持传统的层级文件组织结构，任何有关文件系统的改变都会被记录。数据复制：HDFS将文件分成大小相同的若干份存储于各个数据节点，同时在其它若干数据节点上也保存各个文件块的副本。文件系统元数据持久存储：HDFS文件系统的元数据信息存储在名节点上，名节点将持久记录文件系统上的变化。多副本的流式复制：将名节点上存储副本的节点信息循环写入数据节点。心跳检测和重新复制：数据节点定期向名节点报告状态/读写：HDFS数据文件被分成固定的块，默认块大小为64MB，读/写操作运行在块级。优势:支持大规模文件存储:文件以块为单位进行存储。简化系统设计:简化了存储管理。适合数据备份:每个文件块都可以冗余存储到多个节点上，大大提高了系统的容错性和可用性/hdfs优势：-高容错性(1)数据自动保存多个副本。(2)副本丢失以后，可以自动恢复。适合处理大规模数据 .(1)数据规模:能够处理数据规模达到GB、TB、甚至PB级别:(2)文件规模:能够处理百万规模以上的文件数量，数量相当之大。可构建在廉价机器上，通过多副本机制，提高可靠性/问题：-不适合低延时数据访问，如毫秒级的存储数据无法高效的对大量小文件进行储存(1)储存大量小文件，会占用名节点大量的内存来存储文件目录和块信息。这样是不可取的，因为名节点的内存是有限制的。(2)小文件存储的寻址时间会超过读取时间，违反了HDFS的设计目标。不支持并发写入、文件随机修改(1)一个文件只能由一个写入，不允许多个线程同时写2)仅支持数据追加，不支持文件的随机修改/ceph:无需查表，算算就好,无中心结构，随机写入/**

**分布式数据库：/关系型数据问题无法有效处理大量的半结构化和非结构化数据；无法保证数据并发性、可扩展性和可用性完善的事务机制和高效的查询机制成为负担/非关系型数据库-特点:分布式、非关系型、不保证遵循ACID原则典型代表:HBase、MongoDB、Cassandra、Redis/hbase:一个能提供高可靠性、高性能、面向列、可伸缩、实时读写的分布式数据库，主要用来存储半结构化和非结构化的数据。目标为依靠水平扩展的方式，不断增加廉价的商用服务器集群实现处理非常庞大的数据表，增加计算和存储能力/数据模型：表，行键（主键），列族（表创建时就定义好，访问控制、磁盘和内存的使用统计都是在列族层面上进行的）,列限定符定位列族数据,单元格：前三定位单元格,时间戳：前三加这定位单元格数值/功能模块：-链接到每个客户端的库函数-一个主(Master)服务器：负责管理和维护HBase表的分区信息，监测集群中的分区服务器，确保负载均衡，处理表和列族的创建等-多个分区(Region)服务器：负责存储和维护分配给自己的分区，处理来自客户端的读写请求/客户端并不从主服务器上读取数据，而在获得分区的存储位置信息后，直接从分区服务器上读取数据。并不依赖于主服务器，而是借助于Zookeeper来获得分区的位置信息。因此，大多数客户端从不和主服务器通信，减小了主服务器的负载/分区定位机制：每个分区都有一个RegionId作为唯一标识。一个分区标识符可表示为“表名+开始主键+RegionId”为定位每个分区的位置，需要构建一张位置映射表，表示分区和分区服务器之间的对应关系，元数据表(.META.表)，每行包含两项:分区标识符和分区服务器标识符，当条目过多时分裂成多个分区。根数据表(-ROOT-表)，作为.META.表的分区映射表，记录所有元数据的具体位置，-ROOT-表是不能被再分割的，存在-ROOT-表的唯一分区在程序中被写死。ZooKeeper 中记录 -ROOT-表的存储位置。HBase使用类似B+树索引的三层结构来保存分区位置信息/三级寻址过程客户端访问用户数据之前，需要先访问Zookeeper获取ROOT-表的位置信息。通过访间-ROOT-表获取所请求行所在范围所属的.META表的位置。访问.META.表，获取所需的数据分区的具体位置，客户端直接与管理该分区的分区服务器进行交互。缓存机制：在客户端缓存査询过的分区的位置信息。当缓存失效时，重新发起“三级寻址”过程，并更新缓存/功能组件体系结构：客户端：HBase功能的使用者，包含访问HBase的接口，同时在缓存中维护着已经访间过的分区位置信息。客户端使用HBase的RPC机制与主服务器和分区服务器进行通信。对于管理类操作，客户端与主服务器进行RPC.对于数据读写类操作，客户端与分区服务器进行RPC;ZooKeeper服务器:并非一台单一的机器，可能是由多台机器构成的集群来提供稳定可靠的协同服务。每个分区服务器都需要到xx中注册,实时监控每个分区服务器的状态并通知给主服务器。HBase中可以启动多个主服务器，xx可以帮助选举出一个主服务器作为集群的总管，并保证在任何时刻总有唯一一个主服务器在运行，从而避免Master的“单点失效”问题。ZooKeeper存储-ROOT-表的地址、HMaster的地址HBase表、列族等信息/主服务器(HMaster)每台HRegion服务器均会和HMaster服务器通信，HMaster的主要任务是通知每台HRegion服务器它要维护哪些HRegion。HMaster在功能上主要负责表和HRegion的管理工作，管理用户对表的增加、删除、修改、查询等操作。(1)(2)实现不同分区服务器之间的负载均衡。(3)在分区分裂或合并后负责重新调整分区的分布。对发生故障失效的分区服务器上的分区进行迁移;分区服务器是HBase中最核心的模块，负责维护位于本机的所有分区，并响应用户的读写请求。HBase通常采用HDFS作为底层存储文件系统，来保证可靠稳定的数据存储，因此，分区服务器需要向HDFS读写数据。用户通过一系列HRegion服务器来获取这些数据，一台机器上一般只运行一个HRegion服务器，且每一个区段的HRegion也只会被一个HRegion服务器维护。HRegionServer 内部管理了一系列分区对象HRegion 和一个HLog文件，HLog是磁盘上面的记录文件，它记录着所有的更新操作。每个 HRegion 由多个 HStore 组成，每个 HStore 对应表中的一个列族的存储。HStore是分区服务器的核心，每个HStore包含一个MemStore和者千个StoreFile。MemStore是在内存中的缓存，保存最近更新的数据，StoreFile是磁盘中的文件，StoreFile在底层的实现方式是HDFS文件系统的HFile，HFile 的数据块通常采用压缩方式存储，以减少网络和磁盘I/0;工作原理;读写过程：写：当客户端写入数据时，被分配到相应的分区服务器去执行操作，数据首先被写入到MemStore和HLog中。当操作写入HLog之后，commitO调用才会将其返回给客户端，读：当客户端读取数据时，分区服务器会首先访问MemStore缓存。若数据不在缓存中，则去磁盘上的StoreFile中去寻找;缓存刷新：MemStore缓存的容量有限，系统会周期性地调用Region.flushcacheO 把MemStore缓存里面的内容写到磁盘的StoreFile 文件中，清空缓存，并在HLog文件中写入一个标记，表示缓存中的内容已被写入StoreFile文件，每次缓存刷新操作，都会在磁盘上生成一个新的StoreFile文件，因此，每个Store包含多个StoreFile文件。每个分区服务器在启动的时候会检查自己的HLog文件，确认最近一次执行缓存刷新操作之后是否发生新的写入操作。没有更新，表明所有数据已经被永久保存到磁盘的StoreFile文件中，更新，先将这些更新写入MemStore，然后再刷新缓存并写入到磁盘的StoreFile 文件中，最后删除旧的HLog文件，并开始提供数据访问服务；storefile合并于分区：当访间某个Store中的某个值时，必须查找多个StoreFile，非常耗时，影响查找速度。系统一般会调用Store.compactO将多个StoreFile合并成一个大文件，但由于合并操作比较耗资源，系统只会在StoreFile文件的数量达到一个阈值时才会触发该操作。当多个StoreFile合并后，会逐步形成越来越大的StoreFile文件，当单个StoreFile文件大小超过一定值时，就会触发文件分裂操作。当前的分区会分裂成两个子分区然后下线，新分裂出的两个子分区被主服务器分配到相应的分区服务器上/故障处理数据恢复：HLog文件是一种预写式日志(WriteAheadLog)，更新数据必须首先被记入日志后才能写入MemStore缓存直到MemStore缓存内容对应的日志已经被写入磁盘之后，该缓存内容才会被刷新写入磁盘。Zookeeper 会实时监测每个分区服务器的状态，当某个分区服务器发生故障时，Zookeeper会通知主服务器。主服务器首先会处理该故障分区服务器上遗留的HLog文件。由于一个Region服务器上面可能会维护着多个分区对象，它们共用一个HLog文件，因此该遗密的HLog文件中包含了来自多个分区对象的日志。主服务器根据每条日志记录所属的分区对象对HLog数据进行拆分，拆分结果分别放到相应分区对象的目录下将失效的分区重新分配到可用的分区服务器中，并将与该分区对象相关的部分HLog日志发送给新的分区服务器。分区服务器领取到分配给自己的分区对象及日志后，重做日志中的所有操作，并将日志中的数据写入MemStore缓存，然后刷新到磁盘的StoreFile文件中，完成分区数据恢复。文件设计原则：HBase中设计的每个分区服务器只需要维护一个多分区对象共享的HLog文件。优点:多个分区对象的更新操作所发生的日志修改，只需不断把日志记录追加到单个日志文件中，无需同时打开、写入到多个日志文件，可减少磁盘寻址次数，提高对表的写操作性能。缺点:若分区服务器发生故障，为实现数据恢复，需要将分区服务器上的HLog按照分区对象进行拆分后再分发到新的分区服务器上/hive特点：支持不同的存储类型，如纯文本文件、HBase中的文件可将元数据保存在关系数据库中，减少了在查询过程中执行语义检查的时间。可直接使用存储在Hadoop文件系统中的数据，内置大量用户函数UDF来操作时间、字符串和其他的数据挖掘工具，支持用户扩展UDF函数来完成内置函数无法实现的操作。类SOL的查询方式，可将SQL查询转换为MapReduce的job在Hadoop集群上执行/nosql特点：可扩展性灵活的数据模型与云计算紧密结合；需求（横向扩展,高可用,模式自由）;基石理论：cap一致性:在分布式环境中，多点数据是一致的。Availability可用性:快速获取数据，可以在确定的时间内返回操作结果Tolerance of Network Partition分区容忍性:当出现网络分区的情况下，分离的系统依旧可以正常运行（只可满足二）；base:牺牲一致性，Basically Available(基本可用):允许分区失败。Soft state(软状态):状态可以有一段时间不同步。Eventually consistent:最终数据是一致就可以，而不是每时每刻都一致，分别是响应时间上功能上损失2微观:允许系统中的数据存在中间状态，并认为该状态不影响系统的整体可用性。宏观:允许系统在多个不同节点的数据副本存在数据延时。3因果一致性写一致性会话一致性单调读一致性单调写一致性/四种nosql：优点:便于扩展，适用于云计算环境。与应用程序代码的兼容性好。缺点:数据完整性约束移动至程序。二目前很多键值数据存储系统之间不兼容/文档数据库有点：数据模式的直观性使用JSON,模式的灵活性/使用图数据库来表达现实世界的关系很直接、自然，易于建模。可以很高效的插入大量数据。可以很高效的查询关联数据。提供了针对图检索的查询语言。提供了专业的分析算法、工具。/选择：企业混合式，超市银行关系型/newsql:支持关系数据模型使用SQL作为主要接口**

**5：特征表示将数据转换为有利于后续分析处理形式的一种形式化表示和描述/PCAICA/特征提取,选择:都从原始特征中找出最有效的特征都能帮助减少特征的维度、数据冗余;强调通过特征转换的方式得到一组具有明显物理或统计意义的特征有时能发现更有意义的特征属;从特征集合中挑选一组具有明显物理或统计意义的特征子集能表示出每个特征对于模型性构建的重要性/关联规则反映事物之间的相互依存性和关联性。若两个或者多个事物之间存在一定的关联关系，则其中一个事物能够从其它事物中推断出来/Apriori缺点:候选频繁K项集数量巨大,在验证需要对整个数据库进行扫描，非常耗时,改进方向:通过减少扫描数据库的次数改进I/0的性能(增量式，哈希);改进产生频繁项集的计算性能;寻找有效的并行关联规则算法;引入抽样技术改进生成频繁项集的I/0和计算性能；扩展应用领域。比如展开定量关联规则、泛化关联规则及周期性的关联规则的研究。./kmeans(层次聚类改进，离散点去改进)k小结果较粗糙,每个聚类包含更多的样本导致不同类别样本被划分到同一聚类中;?可能无法充分捕捉数据的内部结构，使得聚类的解释性较差;大聚类结果更细致，每个聚类包含更少的样本，但可能导致过拟合，生成过多的聚类，甚至可能每个样本成为一个独立的聚类;可能增加聚类的复杂性，难以解释和利用；计算复杂度，稳定性（噪声离群点）/mse(均方误差凸函数，易于在优化问题中使用，对大误差特别敏感，突出并减少大误差，缺点异常值敏，输入输出单位不一致/rmse：单位与输入值一致，便于解释+mse,计算复杂/Mae(平均绝对误差)对异常值不敏感，单位同下一样解释，计算;不区分误差大小：所有误差一视同仁，不进行更大惩罚;在优化时不如平滑：在原点不可导,优化问题中不如平滑。/p(Y|X)=(X|y)\*(y)/(x)/GAN:g(生成器),尽量生成真实的图片去欺骗D，D(判别器)尽量把G生成的图片和真实的图片区分开。训练时分别对D和G进行交互迭代,固定G,优化D, 一段时间后,固定D再优化G,直到模型收敛/可视化：观测跟踪数据，分析数据，辅助理解数据，增强数据吸引力;文本可视化河流图，网络（图）时空数据多维数据(散点图平行坐标) /knnK的个数：k小：容易导致过拟合，即模型过于复杂，对训练数据中的噪声和异常值过于敏感。模型具有高方差，结果可能不稳定。计算速度较快。，k大容易导致欠拟合，即模型过于简单，无法捕捉数据的真实结构。模型具有高偏差，结果更稳定但可能不够准确。/如何选取：使用交叉验证方法来选择最优的 𝑘k 值。将数据集分成训练集和验证集，对不同的 𝑘k 值进行训练和验证，选择使验证集上分类精度最高的 𝑘k 值。经验法：用根号n，根据数据集的大小增减k集中式分布：大型主机(Reliability,Availability,Serviceability;高可靠性、高可用性、高服务性可能增加聚类的复杂性，难以解释和利用O吞吐量(同时处理IO通道)(ISA）向后兼容换系统无需测试/人才培养和价格贵,故障即不可用,分布式替代/天河二号神威太湖之光/分布式计算架构：传统并行:集群架构/容错性:共享式(共享内存/共享存储)容错性差;非,好;硬件/价格/扩展性:刀片式服务器,高速网,价格贵,扩展性差;普通PC机,便宜,好;编程/学习难度:难;简单;适用场景:实时细粒度计算密集型;批处理,非实时,数据密集型/mapreduce编程容易,分而治之,计算想数据靠拢：移动数据需要大量网络传输开销;采用了Master/Slave架构，一个Master(运行JobTracker)，若干个Slave(运行TaskTracker)。Hadoop框架是用Java实现的，MapReduce不一定/组成:Client:用户编写的MapReduce程序通过Client提交到JobTracker端;用户可通过Client提供的一些接口查看作业运行状态。JobTracker:负责资源监控和作业调度;监控所有TaskTracker与Job的健康状况，一旦发现失败，就将相应的任务转移到其他节点;会跟踪任务的执行进度、资源使用量等信息，并将这些信息告诉任务调度器（TaskScheduler）而调度器会在资源出现空闲时，选择合适的任务去使用这些资源;TaskTracker会周期性地通过“心跳”将本节点上资源的使用情况和任务的运行进度汇报给JobTracker，同时接收发送过来的命令并执行相应的操作（如启动新任务、杀死任务等）Task 分为Map Task 和Reduce Task两种，均由TaskTracker 启动/split分片：它只包含一些元数据信息:比如数据起始位置、数据长度、数据所在节点等。它的划分方法完全由用户决定；决定map数量；reduce由可用reduce任务槽数目决定/ShuffleReduce：任务通过RPC协议向JobTracker询问Map任务是否已经完成，若完成，则领取数据；Reduce领取数据先放入缓存，来自不同Map机器，先归并，再合并，然后写入磁盘多个溢写文件归并成一个或多个大文件，文件中的键值对是排序的；当数据很少时，不需要溢写到磁盘，直接在缓存中归并，然后输出给Reduce/DeepBench基准:矩阵相乘(密集,卷积(浮点运算,循环池(反馈层，前两个运算组合,allreduce(参数传递或解析/GPU:图形处理器；GPU加速计算:同时利用GPU和CPU，加快科学分析、工程、消费和企业应用程序的运行速度；处理任务不同；CPU由专为顺序串行处理而优化的几个核心组成，GPU则拥有一个由数以千计的更小、更高效的核心(专为同时处理多重任务而设计)组成的大规模并行计算架构;数据并行计算用于数据处理而非数据缓存；延迟高/TPU:张量处理器:低精度计算(深度学习影响小较快计算速度，减少IO操作，相应延迟/FGPA:现场可编程逻辑闸阵列:快速成品，内部反复修改，除错成本低/spark特点：运行速度快:使用DAG执行引擎以支持循环数据流与内存计算。容易使用:.支持使用Scala、Java、Python和R语言进行编程，可以通过Spark Shell进行交互式编程。通用性:Spark提供了完整而强大的技术栈，包括SOL查询、流式计算、机器学习和图算法组件。运行模式多样:可运行于独立的集群模式中，可运行于Hadoop中，也可运行于AmazonEC2等云环境中，并且可以访问HDES、Cassandra、HBase、Hive等多种数据源优点比较hadoop:计算模式不局限于Map和Reduce操作，还提供了多种数据集操作类型编程模型,灵活。提供了内存计算，可将中间结果放到内存中，对于迭代运算效率更高。基于DAG的任务调度执行机制，要优于迭代执行机制/流式数据计算架构：apachestrom/spark/samza三种实时计算系统均为开源的分布式系统，具有低延迟、可扩展和容错性等优点。允许用户在运行数据流代码时，将任务分配到一系列具有容错能力的计算机上并行运行。均提供了简单的API来简化底层实现的复杂程度，**

**大数据安全：指确保数保密性、完整性和可用性，不受到安全威胁影响.禁止用户在没有授权的情况下获取数据;确保数据不被未授权者篡改、损坏、销毁，或在篡改后能够被迅速发现;保证合法用户在需要时可以使用所需的数据，并且数据在传输过程中没有失真 形成原因：传统数据安全防护技术的缺陷(当前，网络攻击攻击效果已经从易察觉的系统宥机、信息泄露转向细小难以察觉的结果偏差;传统的基于监测、预警、响应的安全防护技术难以应对大数据安全问题的动态变化);大数据分布式存储的风险(由于大数据在云端的分布式集中存储和处理，使得安全保密风险也向云端集中;如果云端服务器被攻陷，海量信息可在瞬间被集中窃取);大数据平台安全机制的不足(hadoop为例:用户的身份鉴别和授权访问等安全保障能力比较薄弱;Hadoop中没有原生安全审计功能，需要使用外部附加工具进行日志分析);新型虚拟化网络技术的局限(接口的开放性会引发漏洞暴露和接口滥用的问题;网络功能虚拟化(NFV):部署时通常会外包给第三方虚拟化平台，易发生安全问题);新型高级网络攻击的威胁（例如高级可持续攻击(APT)，攻击者将APT攻击代码长期隐蔽在大数据中；APT攻击的发现难度更大，产严重威胁着网络安全)安全问题分类：大数据平台安全(存储 传输 处理 运行;外部访问控制 基础设施)大数据自身安全(呆障数据源的真实可信性，防止源数据被伪造或刻意制造;保障数据源的可靠性和完整性，尽可能减小数据采集过程中由于人工干预带来误差)大数据应用安全(如隐私问题;利用去标识化、匿名化、密文计算等技术保障个人数据在平台上处理、流转过程中不被泄露;用户应有权利决定自己的信息如何被使用，从而实现用户可控的隐私保护)隐私权的定义：“不受干涉”或“免于侵害”的独处权利;个人享有不被他人干涉和个人不愿被公开的信息被防护的权利。隐私问题变化特征：隐私范围扩大且难以界定;隐私权利归属复杂;隐私保护困难;隐私保护技术：数据隐藏(针对数据挖掘，防范隐私泄露;数据扰动：对数据进行变换，使其中敏感信息被隐藏，在记录之间交换数据的值，在原始数据中随机添加一些噪声;安全多方计算：针对无可信第三方情况，允许多个数据拥有者进行协同计算；输出计算结果，确保各个参与者只能得到既定的输出结果；保证参与者的任何隐私信息不会被泄露。);数据脱敏(:对敏感信息根据脱敏规则,进行数据变形，多次脱敏);数据发布匿名技术(隐藏数据记录与特定个人之间的对应联系。);基于差分隐私的数据发布(保留统计学特征的前提下去除个体特征;中心化差分隐私:数据收集者将数据汇集到第三方数据中心；数据中心进行满足差分隐私的数据扰动；对外发布扰动数据后即可用于统计数据查询；本地化差分隐私由用户在本地进行满足差分隐私的数据扰动再将扰动数据发送给收集者，汇集在第三方数据中心)安全防护技术:数据加密技术(加密阶段:将原始信息经过加密密钥及加密函数转换变成无意义的密文实现信息隐蔽;解密阶段:接收方则将此密文经过解密函数、解密密钥还原成明文。)/数据真实性分析和认证技术(保证大数据的真实可信性，对大数据的发布者进行认证检测;数字签名:通过密码技术对电子文档形成签名;结合了哈希算法等公钥加密技术;保证发送信息的真实性和完整性;加密：发送方用哈希函数从报文文本中生成数字摘要。用发送方的私钥对这个摘要进行加密.加密后的摘要作为报文的数字签名和报文一起发送给接收方，解密：接收方用公钥来对报文附加的数字签名进行解密，再用相同的哈希函数从原始报文中计算报文摘要，两个摘要相同，则接收方就能确认该数字签名是发送方的;数字水印:应用计算机算法嵌入载体文件的防护信息以难以察觉的方式直接嵌入数据载体内部；嵌入过程:将密钥、原始图像和水印信息作为嵌入算法的输入，输出为含水印的图像。提取过程:根据水印的完整性，判断原始数据的完整性；基于数据挖掘的认证技术：收集用户设备数据，分析，确定身份；优势：安全性:攻击者很难再模仿到用户行为；减轻用户负担:避免了由于用户所持有凭证不同而带来的种种不便；认证机制统一:这避免不同系统采用不同认证方式;访问控制技术(属性加密：用一系列属性集来描述用户的身份信息。接收者向服务器进行身份认证时，需要出示与自身属性相关的信任证书。当接收者拥有的属性超过加密者所描述的预设门槛时，用户便可对资源进行解密的，服务器对应的资源发送给接收者；风险：大数据应用系统的复杂性，会存在一些特定的访问需求在设计策略时没有考虑。严格按照预先定义的策略执行访问控制，将产生授权不足无法完成业务的情况，基于风险的访问控制开始衡量访问行为所带来的风险是否为系统可接受的;角色：不是直接授予具体权限给用户在用户集合与权限集合之间建立一个角色集合:构造成用户-角色-权限的授权模型);安全审计技术(规则：利用已知攻击行为特征去比对判断采取相应机制：3步，统计：统计量描述，设定临界值，机器学习：数据挖掘分析，判断趋势并预警);数据溯源技术(标记：标注信息传播，通过目标数据溯源，反向查询：构造反函数查询求逆，适合细粒度数据);APT攻击检测技术(指某组织对特定对象展开的持续有效的攻击活动；检测难度：先进的攻击方法；持续性攻击与隐藏；长期驻留目标系统；应用：网络流量异常，恶源代码，网络安全事件)/**

**行业应用：社会媒体：称为社会化媒体、社会性媒体，指允许人们撰写、分享、评价、讨论、相互沟通的网站和技术；是人们彼此之间用来分享意见、见解、经验和观点的工具和平台，现阶段主要包括博客、论坛、播客等；指互联网上基于用户关系的内容生产与交换平台。社交媒体是人们彼此之间用来分享意见、见解、经验和观点的工具和平台现阶段主要包括Fecebook、twitter、微博、微信、博客、论坛。在互联网的沃土上蓬勃发展，爆发出令人眩目的能量，其传播的信息已成为人们浏览互联网的重要内容，不仅制造了人们社交生活中争相讨论的一个又一个热门话题，还吸引传统媒体争相跟进。/挖掘：事件走势，传播途径，关键词云，观点分析，舆情总结；数据分析以用户（用户识别：在线社交网络可看做异构信息网络，其中的信息通常包括时间、地点、人物、事件等，而用户往往同时存在于多个不同的社交网络中。由于异构的特点，导致同一个人在不同的网络中会呈现一定的差异，如何在此种情况下识别这个人的身份成为研究热点。社群发现：社群是指用户在某段时间内互动形成的具有稳定群体结构、一致行为特征和统一意识形态的个体和社会关系的集合。社群内部用户关系强度强，聚合强度大，而社群之间用户关系强度弱，离散程度大。社群挖掘的目的在于从用户的行为、群体结构和关系模式中发现潜在的规律。社群结构按照用户社会关系和对文本内容的兴趣度划分为两种:·1、以用户个体为中心的社群结构·2、以话题为中心的社群结构），关系(·信息传播用户关系强度的计算源于实际数据的传播模型，它们采用信息本身特性、用户关系、微博网络外部因素等多方面对信息传播进程建模，预测信息传播动态以及用户个体的传播行为。从整体出发，预测信息的扩散速度、范围、广度和深度等;或是从个体出发，预测用户个体传播某条信息的概率，进而研究整个社会网络的信息传播情况。@影响力最大化;影响力计算是针对单个用户节点而言的，而影响力最大化问题涉及网络中的多个用户，考量集体的联合影响力，它利用信息传播模型聚集用户，使用户集合可以最大程度地影响其他用户，从而使信息最大程度地扩散。传统影响力最大化问题新型影响力最大化问题$特征提取与选择收集到的原始文本组织松散，直接用于文本分析会影响分析的准确性。预处理就是采用特征抽取和特征选择的方法将文档组织成固定数目的预定义类别)，内容（话题事件挖掘：事件是指在特定的时间和地点下发生的有前因和后果的事情，而话题是指由所有直接相关事件构成的大事件。话题挖掘的主要任务是话题检测与跟踪，采用历史事件追溯检测和在线新事件自动识别方法，对此已有大量研究，尤其针对完整新闻报导和博客的话题检测已取得了一些成绩。然而，由于微博格式复杂，内容简短，用语不规范等特点，TDT技术不能简单应用到微博；#情感分析情感分析也叫意见挖掘，旨在依据意见目标从语料中识别和提取特定主题的属性、要素和隐含的主观信息。意见目标通常称作实体可以是人物、事件或话题，与要素和子要素相关联，每个要素都有其自己的一套情感属性。微博情感分析可以提取不同领域的公众情绪和意见，可以确定民意调查的影响，有效解释和描述政治事件，预测股票趋势等。）为中心的分析/挑战：数据采集挖掘安全，信息传播效应刻画：xx刻画是一个复杂的问题它受到信息自身社会因素和网络外部因素的综合影响，并且用户本身的属性与信息本身的属性也相互影响，准确全面地反映信息传播效应已成为关键。这一间题的解决还依赖于影响力、用户关系强度和传播规律；影响力计算：基于关系分析的一个具有重要商业价值的研究方向是影响力计算和信息传播的最大化问题。其中信息传播的最大化间题的全局最优化被证明是NP难问题，对于大规模的社会网络，目前只能采用一些优化算法获取近似的较优解，并且对于影响力最大化问题目前的最佳解决算法也只处理了百万级规模的社会网络而目前微博网络节过亿，如何在微博网络中快速计算出固定数量的最有影响力的节点集合还有待进一步探究。特征提取与选择针对传统数据的特征提取与选择方法已有很多，但是不利于处理低频词和发现新特征，而这种情况在微博数据中大量存在。与词频模型相比，序列模式挖掘保持了词的顺序并可以捕捉潜在的语义，更能解释话题。但是采用模式挖掘的两大挑战是:大量冗余模式的产生和长模式的低支持度问题/微博新闻挖掘:如何在线实时处理这种社会化的短文本流?如何识别新闻话题?如何实时检测新闻事件?如何判断事件的连续性?如何挖掘这种动态的关联演化性?如何从海量博文中提取有意义且易理解的微博话题?挖掘到的新闻以什么形式呈现?如何设计针对微博的动态新闻集成系统/社会媒体大数据融合随着社会网络服务的发展，用户在社交互动中加入了多种服务，并收集了大量的信息。因此，如何整合分布式社会网络，进而对各种社会媒体数据源进行融合，为知识的挖掘提供更好的数据资源已经成为亟待解决的问题/跨语言情感分析挖掘情感是为了体现商业价值，目前大数据向跨语言融合迈进，相应的情感分析也向跨语言情感分析发展。但是，语言的不同体现在语言特征、要素分布的不同，语言间关联的障碍使得跨语言情感分析成为更大的挑战，这是目前亟待解决的问题。**

**时间戳优点实现简单：基于时间戳的增量抽取逻辑相对简单，只需记录上次抽取的时间戳，下一次抽取时提取大于该时间戳的数据。轻量级：对数据库性能影响较小，因为不需要额外的触发器或复杂的日志处理。通用性强：适用于大多数数据库和数据源，不依赖于数据库的具体实现。缺点时钟同步问题：依赖数据库服务器的系统时间，如果时间不同步，可能导致数据遗漏或重复。修改时间字段依赖：需要每个表中有可靠的更新时间字段，如果表中没有时间戳字段或时间戳字段不准确，将无法使用这种方法。误差和延迟：如果数据更新频率高，可能会有一些数据更新在边界时间戳附近被遗漏。日志优点精确性高：能够捕捉所有数据变更，确保增量数据的准确性和完整性。无遗漏：可以确保所有插入、更新和删除操作都被记录下来，无论数据量多大或更新频率多高。回溯能力：某些日志系统允许回溯到特定时间点，方便数据恢复和重建。缺点实现复杂：需要对数据库的日志系统进行配置和管理，对一些数据库可能需要特殊的支持或工具。性能影响：日志记录和读取可能会对数据库性能造成一定影响，尤其是在高并发、高更新频率的系统中。存储开销：日志文件可能会占用大量存储空间，需要定期清理和维护。 触发器优点实时性高：能够在数据变更时立即触发数据抽取操作，确保数据的实时性。细粒度控制：可以精确控制哪些操作需要被捕获，适用于复杂的数据抽取逻辑。独立性：不依赖于数据库的日志系统，可以独立实现增量抽取逻辑。缺点实现复杂：编写和维护触发器代码可能比较复杂，尤其是在需要处理大量表和多种操作时。性能影响：触发器在每次数据变更时都会执行，可能会显著影响数据库性能，尤其是在高并发环境中。维护成本高：触发器代码需要和业务逻辑紧密结合，任何数据结构或业务逻辑的变化都需要同步更新触发器代码。选择建议时间戳方法：适用于数据更新频率较低、系统时间同步可靠的场景，且数据库表中有可靠的更新时间字段。日志方法（CDC）：适用于需要高精度、无遗漏的数据抽取，能够接受一定实现复杂度和性能影响的场景。触发器方法：适用于需要实时数据同步、能够承受一定性能影响，并且有能力维护复杂触发器逻辑的场景。**