中科院计算培训中心

Part 8 流数据挖掘相关技术

流数据处理模型

- 杨文川

1）流数据挖掘及分析
2）Storm和流数据处理模型
3）流处理中的数据抽样
4）流过滤和Bloom filter

流数据

- 数据以一个或多个流的方式到来
  - 如果不对数据进行及时的处理或者存储，数据将会永远丢失。
  - 假定数据到来的速度实在是太快，以致将全部数据存在活动存储器(即传统数据库)，并在选定的时间进行交互，是不可能的。

流汇总

- 数据流处理的每个算法，都在某种程度上包含流的汇总(summarization)过程。
  - 首先考虑如何从流中抽取有用样本，以及如何从流中，过滤除大部分“不想要”的元素。
  - 然后，展示如何估计流中的独立元素个数，其中估计方法所用的存储开销，远少于所有元素的开销

窗口

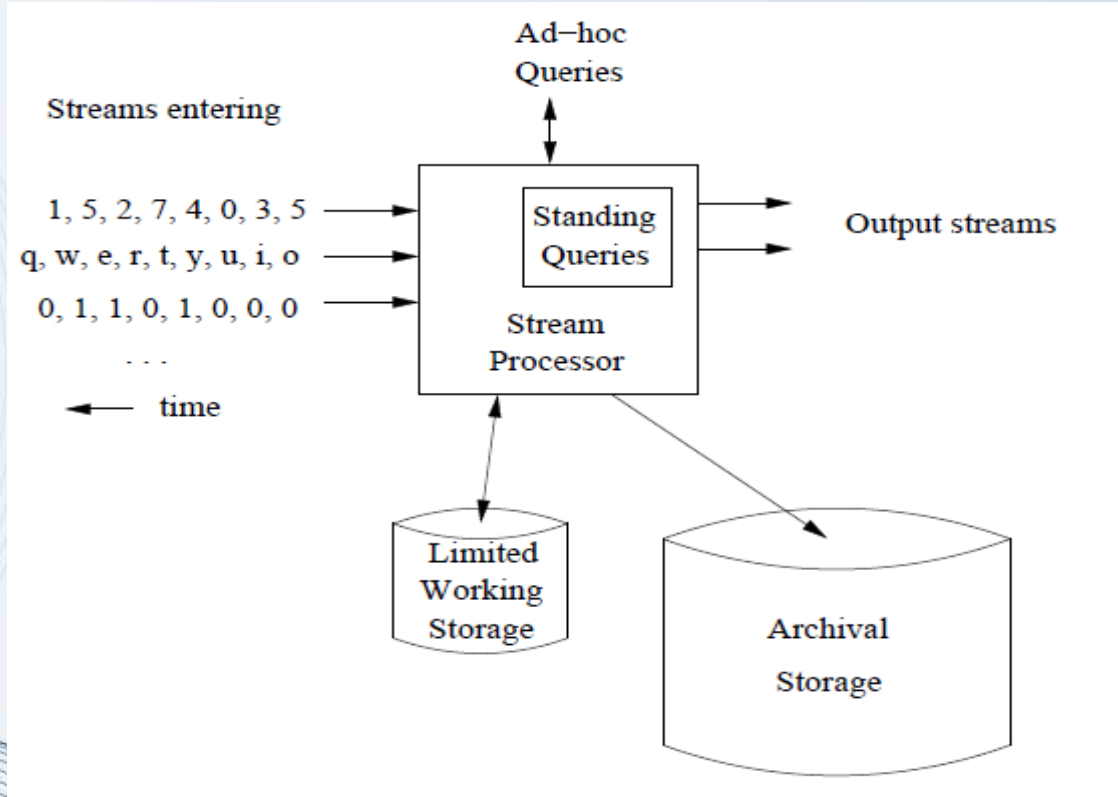
- 另外一种对流进行汇总的方法是
  - 只观察一个定长“窗口”，该窗口由最近的n个元素组成，其中n是某个给定值，通常较大
  - 然后就当它是数据库的一个关系一样，对窗口进行查询处理。
- 注意
  - 如果有很多流并且/或者n很大，可能无法存下每个流的整个窗口。
  - 即使对这些“窗口”都需要进行汇总处理。

流数据模型

- 首先讨论流中的元素及流处理过程。
- 接下来解释流和数据库的区别，以及在处理流时遇到的特殊问题。
- 最后将考察流模型的一些典型应用

一个数据流管理系统

- 流处理器，可看成是一种数据管理系统，系统的高层组织结构如图。



- 若干数量的流进入系统，每个流可以按照各自的时间表来提供元素
- 不同流的数据率，或数据类型可能不相同，流中元素到达的时间间隔，未必满足均匀分布。
  - 流元素的到达速率，不受系统的控制，这是流处理和DBMS中数据处理不同之处。
  - DBMS控制数据从磁盘读出的速率，不用担心执行查询时，会有数据丢失。

数据流管理关键组件

- **大容量归档存储器Archive Storage**
  - 流可以在大容量归档存储器上，进行归档处理
  - 在归档存储器上，一般不能对查询进行应答查询
  - 只有在特殊情况下，才可以使用耗时的检索过程来处理查询。
- **工作存储器Working Storage**
  - 流汇总数据，或者部分流数据，可以存在工作存储器上，该存储器可以用于查询应答的处理。
  - 工作存储器可以是磁盘或者是内存，这取决于查询处理的速度需求。

流数据源的例子

- 不管采用哪种存储介质，其容量都是有限的，不能存储所有的流数据中所有数据
- 先考虑一些自然出现的流数据类型，包括：
  - 海洋传感器数据流
  - 卫星图像数据流
  - 互联网及Web流量
  - Web网站收到的数据流

传感器数据流

- 海表面的高度变化异常迅速，传感器可以每秒10 次的频率，传回海表高度和温度数据。
  - 如果每次传送的是4字节数据，那么一个传感器每天产生的数据量为3.5MB。
  - 为探索海洋，假设每150平方英里海洋部署1个传感器，每个传感器都以每秒10次的速率传回数据
  - 环美洲的海洋需要100万个传感器，每天传回的数据就有3.5TB，全球海洋面积3.6亿平方海里，需要360万个，每天传回10T以上的数据
  - 需要考虑，哪些数据要存放在工作存储器、哪些数据只能放在归档存储器中

互联网及Web流量

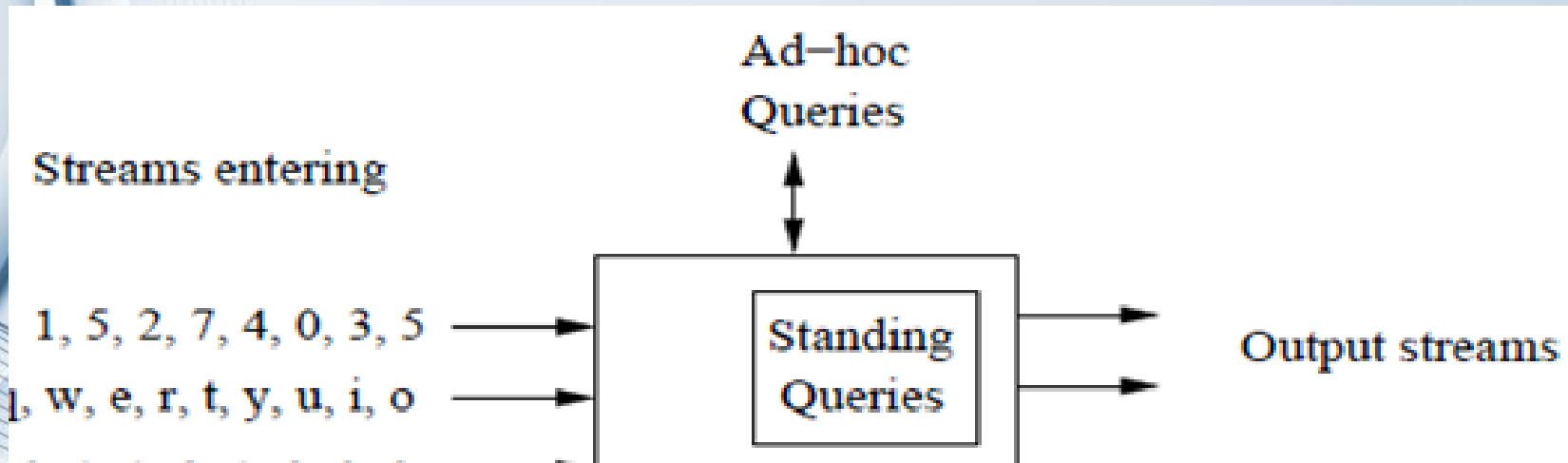
- 互联网当中的交换节点，从很多输入源接收IP包流，并将其路由到输出目标。
  - 交换机的任务主要是传输数据，而非保留或查询数据。
  - 但是将更多功能放入交换机已经成为一种趋势，比如DDOS攻击的探测能力，或者基于网络的拥塞信息，重新对包进行路由的能力。

Web网站收到的流

- 包括各种类型。
  - 例如，谷歌一天收到几亿个搜索查询。
  - 雅虎的各个不同网站上收到数十亿个“点击”。
- 基于这些流数据，可发现很多有趣的结果
  - 比如，“咽喉痛或咽喉炎”之类的查询频次的上升，能够让对病毒的传播进行跟踪
  - 某个链接的点击率的突然上升，可能意味着有些新闻连向此网页，否则的话可能意味着，该链接失效急需修复

流查询

- 对流进行查询主要有两种方式。
- 一种称为固定查询 Standing Query
  - 固定查询基本不变的执行，并在适当的时候产生输出结果。
- 另一种查询称为 Ad hoc查询，



固定查询的例子

- 海表面温度传感器产生的流数据上，可能有这样一个固定查询，
  - 即当温度超过25摄氏度时输出警报。
  - 由于该查询仅依赖于最近的那个流元素，因此对它进行处理相当容易。

其它的固定查询

- 当一个新的温度读数到达时，输出最近24次读数的平均值。
  - 如果存储了最近24个流元素，那么对上述查询的处理也比较容易。
- 当新的流元素到达时，处于排名第25位的流元素，对于上述查询的处理不再有用
  - 因此将它从工作存储器中去掉
- 查询迄今为止传感器所记录的最高温度
  - 需要维护一个最高温度值

Ad hoc查询

- Ad hoc对当前某个或多个流仅提交一次查询。
  - 如果没有存储所有流数据，不能做到这一点，
  - 也不能指望系统，能够应答关于流的任意查询。

保存每个流的滑动窗口

- 若希望询问的Ad hoc查询类型较广，一种办法是，在工作存储器上保存每个流的滑动窗口(sliding window)。
  - 滑动窗口，可以是最近到达的*n*个流元素，也可是在最近*t*个时间 (如一天)到达的所有元素。
  - 如果将每个流元素作为一个元组，就可把窗口看成是DBMS，而在其上执行任意的SQL查询
  - 流管理系统必须在新元素到达时，删除最早的那些元素，保持窗口的新鲜度

流处理中的若干问题

- 首先，流元素的分发速度通常很快。
  - 必须要对元素进行实时处理，否则就会永远失去处理它们的机会，除非访问归档存储器。
  - 流处理算法通常在内存中执行，一般不会(或极少)访问二级存储器，这一点相当重要

需要新技术

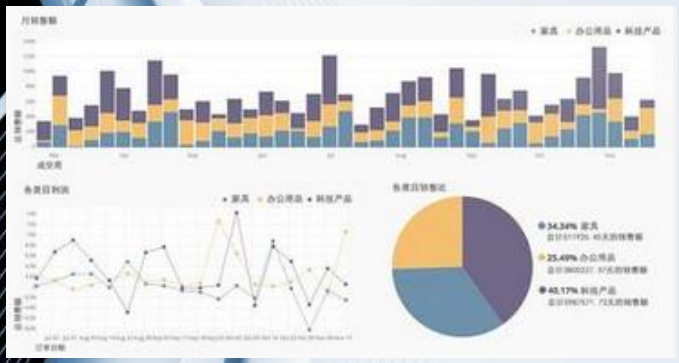
- 总之，当内存足够大时，流数据的处理很容易解决
  - 但要在一个真实规模的机器上，获得现实的处理速度，内存问题就变得相当困难，需要引入新技术来解决
- 即使当数据流很慢(如海洋传感器数据)时，也可能存在多个这样的数据流。
  - 即使每个流本身处理所需内存很小，但所有数据流的内存需求相加，很容易超过内存的可用容量

Storm的诞生

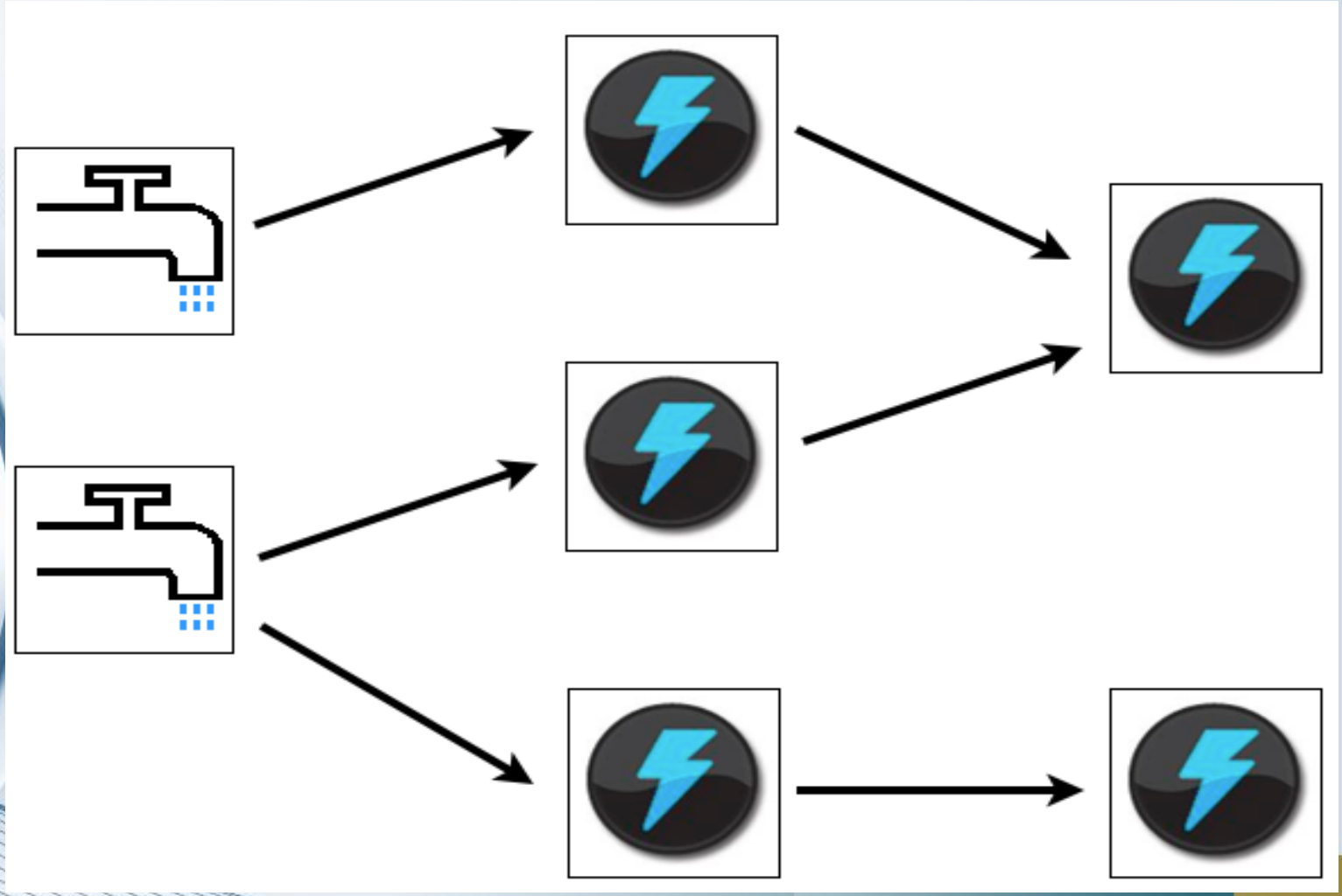
- Storm是随着实时大数据处理的需求而生的
  - 最早用在Twitter上，在分布式的环境下，不间断地实时处理数据。
  - 在分布获得了极大的成功后，由Twitter开源公开
- Storm成为处理实时大数据的最实用工具之一。

流处理工具Storm

- Storm能轻松可靠地处理无界的数据流，像Hadoop批处理一样对数据进行实时处理
  - Storm能用到很多场景中，包括实时分析、在线机器学习、连续计算、分布式RPC、ETL等。
  - Storm的处理速度非常快，每个节点每秒钟可处理100万条消息。
  - Storm是可伸缩的、容错的，能保证数据根据的设定被妥善处理，便于进行设置和操作。



Storm的抽象示意图



Storm的核心技术和基本组成

- Storm框架的核心由7个部分组成
- 最重要的是
  - Topology(拓扑)
  - Stream (流)
  - Spout(喷口)
  - Bolt(螺栓)

两个一般化的结论

- 流处理算法，有两个一般化的结论
  - 获得问题近似解，比精确解要更加高效。
  - 采用哈希相关技术，被证明很非常有效
    - 为产生与精确解相当接近的近似解，哈希技术将随机性技术，引入到算法中。

流当中的数据抽样

- 作为流数据管理的例子，将考察流中的可靠样本抽取问题。
  - 同很多流算法一样，该抽取“技巧”中包含了哈希方法的使用

流过滤

- 常见的流数据处理方式是**选择，或过滤**。
  - 只接受流当中，满足某个准则的元组集合
- 被接受的元组，会以流的方式传递给另一个过程，而其他元组被忽略。
  - 如果选择的准则，基于元组的某可计算属性得到(如第一字段小于10)，那么选择很容易完成
  - 当选择准则中，包含集合元素的查找时，问题就变得复杂。特别当集合大到无法在内存存放时
    - 采用布隆过滤技术，可去掉不满足选择准则的大部分元组

垃圾邮件例子

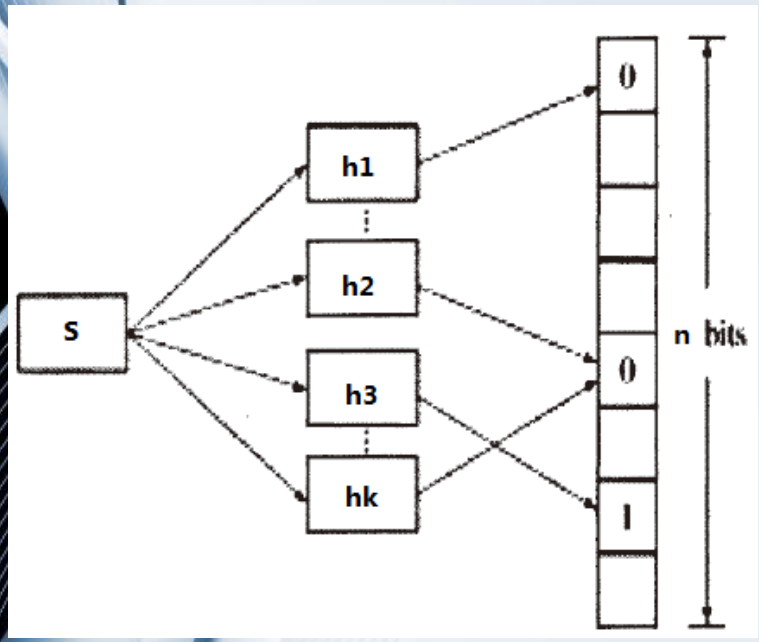
- 假定集合S中包含10亿个允许的邮件地址，确信这些地址不是垃圾邮件地址。
  - 流数据由邮件地址及邮件本身，组成二元组构成
- 典型的邮件地址为20或更多字节，将S保存在内存中是不合理的。
  - 要么基于磁盘访问，来确定是否让任何给定的流元素通过，
  - 要么设计一种办法，能过滤掉大部分不想要的流元素，且该方法所需内存大小，低于可用内存
  - 假定有1GB的可用内存

- 布隆过滤技术中，内存被当成位数组来使用
  - 这种情况下，由于1个字节有8位，所以1GB内存可以容纳80亿个位。
  - 可设计一哈希函数h，将邮件地址映射到80亿个桶中。
  - 这时将S中的每个元素映射到某位，并将该位设置为1，而数组中所有其他的位仍为0

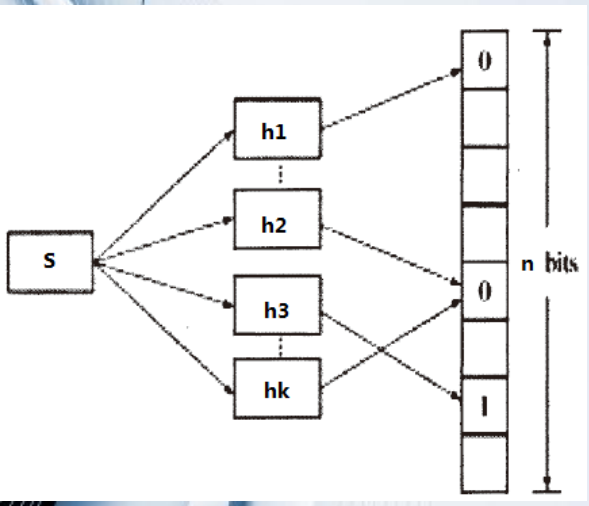
- 由于*S*中有10亿个元素，因此所有位当中有近1/8的位为1。
  - 哈希函数可能将*S*的两个元素，映射到同一个位，因此确切为1的位所占的比例会略低于1/8。
- 当一个流元素到达时，对其邮件地址进行哈希操作
  - 如果该邮件地址哈希之后对应的位为1，那么就让邮件通过
  - 但若对应的位为0，则可以确信该邮件地址不属于S，从而丢弃该流元素。

- 会有一些垃圾邮件地址也会通过。
  - 邮件地址不在*S*中的流元素中，大约有1/8会被哈希到位1从而通过过滤。
- 由于大部分邮件都是垃圾邮件(大约80%)，因此剔除7/8的垃圾邮件，得到的好处很大。
  - 如果想剔除所有垃圾邮件，只需检查通过了过滤的，那些邮件的地址是否真正属于S。
  - 这些检查过程，要使用二级存储器来访问S本身，
  - 例如，可以将多个过滤器串联起来使用

布隆过滤器



- 布隆过滤器由几部分组成：
  - (1)n个位组成的数组，每个位的初始值都为0
  - (2)一系列哈希函数$h_1,h_2,\ldots,h_k$组成的集合。
  - 每个哈希函数将“键”值映射到n个桶(对应位数组中n位)中。
  - (3)m个键值组成的集合S
- 布隆过滤器的目的，是让所有键值在S中的流元素通过，而阻挡大部分键值不在S中的流元素



- 位数组的所有位的初始值为0。
  - 对S中的每个键值K，利用每个哈希函数进行处理
  - 对于一些哈希函数hi及S中的键值K，将每个$h_i(K)$对应的位置为1。

当键值为K的流元素到达时

- 检查所有的$h_1(K),h_2(K),\ldots,h_k(K)$对应的位是否全部为1
- 如果是，则允许该流元素通过，
- 如果有一位或多位为0，则认为K不可能在S中，拒绝该流元素通过

谢　谢