**主要内容（大概）**

**构造用于高效窗口查询处理的紧凑时间序列索引**

**Constructing Compact Time Series Index for**

**Efficient Window Query Processing**

◎介绍了一种时序数据分析的算法Winldx，用于高效的窗口查询处理，主要功能就是将高相似度的窗口进行集群，然后对集群内的窗口形成紧凑的索引结构，从而加快查询的效率。这种算法与已有的最先进时间序列索引算法相比，仍旧具有一定优越性。（关键词：相似性检测、集群，对于数据异常或许能起到一定参考作用，比如几乎没法跟其他子序列分到同一个集群就可能是异常。所以或许，如果iotdb用类似的方法构建索引，可不可以从这个过程监测异常值的出现呢？）

主要内容１：wss问题

·问题：wss问题，描述的是怎么在有限的时间窗口内找到最相似的数据。

·问题描述：给定一个数据集D= {...}，里面存放的是一长串时序数据，给定一个任意的时间窗大小W，给定一个时间长度为W的查询序列Q，要求在D中找到这样的Q的K近邻。

·问题应用：在不同的时间段搜索关键字“FIFA”，所呈现的头几条检索结果也是不一样的，因为不同时间下不同关键词与“FIFA”的相关程度是不一样的。采用ｗｓｓ查询能够在一个局部时间内找到所期望的最近邻，而无需对整个长时间序列数据进行分析。

主要内容２：窗口相似性判定和窗口聚类

·总体聚类流程：首先，为了度量时间序列之间的关系，用invSAX表示时间序列，对时间序列进行排序并分割成子集。其次，我们根据子集的相似性将窗口聚类成组。第三，基于子集，在每个聚类中建立统一的索引。

·相似度用距离来表示，根据invSAX和序列排序来定义窗口之间的距离。在窗口W中，序列关系由id子集的序列表示，之后将ｉｄ序列拆分为一定数量的不相交子集，然后定义了一个用交集基数控制的距离公式，称为ｉｄ距离，表征窗口ｗ中子序列之间的相似程度（就是看相同的ｉｄ是多是少）。

·窗口聚类：我们的方法是一种迭代精化过程，类似于k-means中使用的方法。通过迭代，它将窗口聚类为K组，C={C1，C2，··，CK}，这使距离之和最小化；通过一定手段定义聚类质心μ，以便1）它可以表示集群中窗口的典型id表示，2）它可以使迭代过程收敛。

主要内容3：索引构建

·为了支持W中的查询，我们根据窗口距离将W中的所有窗口分组到不同的集群中。对于每个集群Ci，将构建一个紧凑的索引（即WTreei）来支持属于Ci的窗口的所有查询。WTreei是一个由两个核心组件组成的二进制树：序列组织和SAX信封（论文提及）。具体构建方式似乎很复杂。

主要内容４：应用

·对于窗口W上的任何查询Q，我们首先计算归一化查询序列的PAA索引，表示为PAA（ξQ）。然后我们定位W所属的簇Ci。查询处理根据∑Ci（拆分后的Ｃｉ的ｉｄ子集）和WTreei检索结果。查询的一般思想是在WTree中从根节点到叶节点进行搜索。WTree的结构对于修剪子树具有良好的性质。此外，∑Cj的子集只包含序列id，而不是原始数据。一旦定位了目标叶节点，我们的方法就使用∑Ci的相应子集中的序列id来定位数据序列的偏移量，并从磁盘中检索原始数据。

**用于高效存储的时间序列数据编码：Apache IoTDB中的比较分析**

**Time Series Data Encoding for Efficient Storage: A Comparative**

**Analysis in Apache IoTDB**

◎介绍了面对不同特征数据，各种不同的数据编码方式的优劣，提出了基本的数字数据编码算法和字符类数据编码算法，提出了搭载编码格式推荐模型的未来设想。只需要注意iotdb中的数据不一定都采用相同的编码方式。

主要内容１：可能影响编码性能的几个时间序列数据特征

数据规模对数据编码的影响非常大，要求更多的编码位。选择均值、方差、极差作为其尺度的衡量。

δ特征反映数据的波动幅度，这对时序数据十分重要。使用均值（DS）、方差（DS）和极差（DS）来评估δ可能多大。

重复度：在记录数据时不冗余存储重复的数据，而是记录下重复的次数。我们只对重复8次以上的

增量：相邻记录值差可能不为0（非重复），如果所有的差符号都是正的，换句话说，时间序列值总是在增加，则编码性能更好。相反，当差值为负时，即，减小时，编码性能将是糟糕的。从这个意义上说，我们将Count（IS）定义为具有相邻时间戳的递增值的数量对增量进行考量。

数据类型：不同的数据类型同样会影响编码效果。

主要内容2：数字数据编码

·基于差分的编码方式。（尽可能减小编码的绝对值）参考上述对无损需求和系统架构的讨论，我们介绍了六种适合在Apache IoTDB中实现的编码算法，包括TS\_2DIFF[8]、GORILLA[37]、SPRINTZ[20]、RLE[27]、RLBE[41]和RAKE[21]。前两者比较常用，效果好一些。

·基于游程长度的编码方式。（减少相邻重复值的空间成本）加入了比特打包技术和RAKE技术之后的传统编码算法改良版有更加有效的进步。

·混合编码

主要内容3：文档数据编码

。。。（还提到了传统的编码算法和实验评估，感觉没什么帮助）

**iotdb中，不同的编码算法有利于不同的数据特征。它促使我们根据不同数据集的特征推荐不同的编码算法，预计会利用先进的机器学习模型对某一批输入的数据推荐最优的编码方案。iotdb内的数据是以多种编码方式存储的。**

**Apache IoTDB：物联网时间序列数据库**

**Apache IoTDB: Time-series Database for Internet of Things**

◎一篇报告，只有4页，对iotdb的一个总体简要介绍。说明了存储数据采用的Tsfile格式文件、语言、运行兼容等基本信息。由于我们用不上分布式集群，所以感觉好像看不看没啥问题，其实看了也没有用。

对IoTDB的介绍。

逻辑上，IoTDB对时间序列数据点描述为这种形式：< device,sensor,timestamp,value >.IoTDB的设计选择将数据存储在一个开放的本地时间序列文件格式中，既可以使用查询/存储引擎访问数据库，也可以使用Hadoop/Spark访问单个数据副本。IoTDB提供了一种类似sql的语言、本机API和restful API来访问数据。

TsFile是IoTDB中存储时间序列数据的主要数据文件格式。TsFile主要由两部分组成:数据内容(chunk, Pages)和索引。每个块存储一个时间序列在一定时间范围内的数据。在块中，数据被分成几个页面，这些页面是磁盘上数据存储的基本单元。每个页面将数据点存储在一对列中，即时间戳和值。时间戳由二阶差分编码，值字段支持bitmap, Gorilla, RLE等压缩算法，以节省磁盘空间。Snappy还用于对历史数据进行高级压缩。每个时间序列块中的数据在TsFile中按时间排序。

为了进一步提高查询性能，每个序列的max/min时间、max/min值、数据点的计数和平均值都是预先计算出来的，并以Summary Info的形式存储在文件、块和页面的级别上，从而大大加快了过滤和聚合操作。

其他还包括iotdb如何处理查询请求、如何抽取数据等的说明，以及优势分析等。