面向 AIS 数据的分布式空间数据仓库

Alex S. Klitgaard Lau E. Josefsen Mikael V. Mikkelsen Kristian Torp 奥尔堡大学 计算机科学系 丹麦奥尔堡
alex.klitgaard@gmail.com,lau@josefsens.dk,mvmi@cs.aau.dk,torp@cs.aau.dk

摘要

来自船舶的 AIS 数据非常适合分析单艘船舶的移动情况，以及监测特定区域内的所有船舶。然而，AIS 数据在投入使用前需要经过清洗、处理和存储。本文介绍了一套系统，该系统包含一个高效、模块化的 ETL 流程，用于加载 AIS 数据，并构建了一个分布式空间数据仓库，用于存储船舶轨迹。为高效分析大量船舶数据，本文提出了一种采用栅格方法查询 AIS 数据的方案。本文设计、开发并评估了一个采用细粒度单元表示和热力图展示的空间分区数据仓库。目前，该数据仓库存储着 3.12 亿公里的船舶轨迹，最大的数据表包含超过 80 亿行。研究发现，搜索单元表示比搜索轨迹表示更快。此外，我们还表明，空间划分分片使大范围内的单元分析和热力图分析均能实现稳定的良好扩展性：当工作节点数量增加 5 倍时，性能提升幅度为 354% 至 1164%。

关键词

时空、空间分区、AIS空间分布、轨迹、移动对象、单元表示、热力图

1 引言

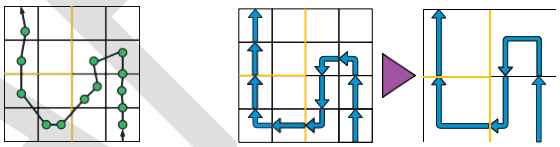
从船舶收集的自动识别系统（AIS）数据在分析方面展现出良好前景 [16]。例如，26th 月发生的北溪水下天然气管线爆炸事件 2022 [2] 表明，可以利用 AIS 数据确定爆炸发生时哪些船舶位于天然气管线附近

尽管 AIS 数据在分析方面展现出巨大潜力，但必须充分注意克服其固有局限性 [16]。AIS 数据的固有局限之一是数据质量不佳，因为该协议最初旨在帮助提高海上安全。因此，在使用 AIS 数据进行分析之前，有必要先清洗数据

基于对船舶领域的了解。丹麦 AIS 数据
已存储了近二十年，因此积累了大量可用于分析的 AIS 数据，例如交通模式挖掘[34] 和吃水图 [28]。

现有 AIS 数据量庞大，无法在单台机器上存储和处理。因此，为了高效处理轨迹数据而提出的系统（例如 [9, 35],）会使用分布式引擎，将数据存储在工作节点集群上。

DIPAAL平台由一个模块化 ETL 流程构成，负责清理和转换 AIS 数据，然后将数据加载到基于 PostgreSQL 的数据仓库（DW）中。如图 1 所示，数据仓库支持 AIS 数据的轨迹表示和网格表示。



(a) 轨迹 **(b) 带汇总的单元格**

图 1：采用空间划分的 DIPAAL 中的 AIS 数据

轨迹是按时间顺序排列的 AIS 点集合，如图 1a 所示，其中绿色圆圈代表单个 AIS 数据点。网格则是用于聚合轨迹信息的空间区域，如图 1b 所示。单元表示由一个层次结构构成，用于存储多个粒度下生成的网格。Citus PostgreSQL 扩展通过将 DIPAAL 数据仓库中的数据分片到工作节点集群中，实现数据分布。我们采用了一种新颖的空间分布方法，以便对网格和热力图表示进行空间聚合。该方法创建空间划分，将网格和热力图数据分布到所有可用的工作节点上。这些空间划分采用 kd 树方法创建，并以整整一年的 AIS 数据为依据。

空间划分的一个示例如图 1 中的黄色线条所示：右侧有一个较大的划分区域，左侧有两个较小的划分区域。从图 1a 和图 1b 可以明显看出，单元表示比轨迹表示更适合空间分布，因为轨迹可能跨越多个划分区域，而网格则完全位于单个划分区域内。

本文探讨基于 kd树的空间分布方法如何影响不同查询类型的查询运行时间以及底层硬件的利用率。此外，本文还评估了在不同面积的查询中，从单工作节点配置扩展至五工作节点配置时，该空间分布方法的扩展能力。

DRAFT

本文的贡献如下：

- 实现一种空间分布方法，通过确保工作节点上的空间数据局部性，并支持将空间聚合下推至工作节点，从而缩短基于单元表示的查询运行时间。 - 按需生成用于分析的分布式热力图 - 对 DIPAAL 平台进行广泛评估，深入了解平台的查询运行时间表现，以及其从一个工作节点扩展至五个工作节点的能力。 - 通过加载、存储并分析五年的 AIS 数据，展示 DIPAAL 设计的可行性。

目前，数据仓库中已加载5年的数据，包含1,390万条轨迹，共计4.14亿公里的船舶轨迹，覆盖1,111,143天。观测到57,238个唯一船舶标识符（海上移动业务识别码）。最大的关系包含100亿行。

本文其余部分的结构如下：第2节介绍本文使用的定义以及 DIPAAL 的架构。第3节阐述 ETL 设计和数据加载流程。第4节介绍数据仓库的结构及其设计选择。第5节讨论空间划分方法。第6节介绍实现。第7节评估 DIPAAL 的效率。第8节介绍相关工作。 最后，第9节总结全文。

2 架构

本节介绍 DIPAAL 的架构。首先给出关键定义，然后概述整体系统架构，最后介绍数据仓库设计的相关细节。

2.1 定义

定义 1. 点：点 $p = \{t, lng, lat, MMSI, ais\}$，其中 t 是创建该点的时间戳，lng 和 lat 分别是经度和纬度坐标的缩写。*MMSI* 指发射船舶的海上移动业务标识，而 ais 是其他可选的 *AIS* 属性，例如吃水和船舶类型。

MMSI 是一个 9 位识别号码，必须用于识别 [12]。定义 1 中的其他可选属性 *ais*，是 AIS 消息数据中与船舶相关的属性。例如对地速度（SOG）、目的地、航向和航行状态，正如 Bereta 等人 [12] 所述。

定义 2。 轨迹：轨迹 T 是按时间戳升序排列的点序列 $T_t = \{p_1, p_2, ..., p_n\}$，其中 $n \geq 2 \wedge p_1 。 t < p_2 。 t < ..... < p_n 。$ 。

在 DIPAAL 中，轨迹必须至少包含两个点，否则无法表示随时间发生的移动。轨迹中的点必须按时间顺序排列。

定义 1 和定义 2 与数据源无关，适用于大多数（如果不是全部）AIS 数据集。

由于 DIPAAL 同时使用分片和分区技术，因此有必要明确区分分片与分区。

定义 3。 分片：关系被划分为多个分片，以实现数据库的水平扩展。每个分片都是父关系的一个子集，并且可能位于分布式数据库中的不同工作节点上。这称为分片 *[30]*。

定义 4。 分区：分区是指在同一个工作节点上，将一个逻辑关系拆分为多个更小的物理关系*[19]*。



图 2：DIPAAL 架构

2.2 平台

图 2 展示了 DIPAAL 的架构，其中紫色箭头表示数据处理，绿色箭头表示数据请求，蓝色箭头表示响应。圆柱体表示数据存储，方框表示软件组件。

为提升数据质量，我们定义了一组清洗规则。ETL 流程遵循 Nielsen 等人 [28] 定义的规则。这些清洗规则与 Graser [18] 的工作类似。

数据仓库存储经 ETL 流程处理的 AIS 数据。因此，系统提供了多种表示形式。根据定义 2，粗粒度对应网格，最细粒度为轨迹，如图 1 所示。

高级用户可以直接查询数据仓库。其他用户可以向预先优化的端点发送 REST API 请求。API 可通过 https://dipaal.dk 访问。API 生成的图像和栅格示例可在 https://dipaal.github.io 查看

2.3 数据仓库

底层存储引擎必须支持时空数据和分布式存储，才能为大规模轨迹数据创建高效的查询平台。PostgreSQL 通过扩展支持时空数据和分布式存储。DIPAAL 使用扩展 Citus [3] 实现数据分布，并使用 PostGIS [5] 和 Mobili- tyDB [4] 提供时空支持。

分布式存储支持 Citus 通过对关系（定义 3）进行分片，并将这些分片存储在工作节点集群中，使 PostgreSQL 能够实现数据分布。Citus 支持共置关系 [14]，，也就是说，分布属性值相同的行位于同一个工作节点上。共置可提高数据局部性，并支持本地连接 [14]，，从而显著缩短查询运行时间。相比之下，非共置连接必须在工作节点之间传输数据才能执行连接。

数据分布还使 Citus 集群中的工作节点能够分担工作负载：每个工作节点使用本地数据计算查询结果的一部分，再将这些部分合并并呈现给用户 [13]。

时空支持 PostGIS 扩展了 PostgreSQL，提供了一系列空间数据类型、运算符和函数。PostGIS 用于栅格操作，以实现 DIPAAL 的热力图功能，并提供空间连接条件。

DRAFT

MobilityDB 基于 PostGIS 扩展构建，通过提升 PostgreSQL 和 PostGIS 提供的运算和函数，增加了对时间数据和时空数据的支持 [10]。MobilityDB 通过定义多个类型构造器，既支持处理离散时空数据，也提供移动对象数据库（MOD）功能 [10]。

3 ETL 设计

ETL 流程由五个可互换且彼此独立的模块组成，以便实现关注点分离并提高模块化程度。该流程包含五个主要步骤，其中介绍的前三个步骤与数据仓库无关。这些部分可以在多台机器上并行执行，无需协调。下文将详细介绍各个模块。这里介绍一个基本用例，其中以下步骤按同步方式执行。

文件下载模块 该模块提供了一个轻量级集成，使 ETL 流程能够自动查找、下载并准备原始 AIS 数据源文件。例如，可对归档文件进行解压。

数据清洗模块 该模块通过实施 [18, 28] 定义的特定领域清洗规则，剔除有问题的 AIS 数据。主要工作包括移除维度异常的船舶、海上移动业务标识错误的船舶，以及位置位于陆地上的船舶。

轨迹构建模块 轨迹构建模块 负责构建存储在数据仓库中的轨迹，参见定义 2。

该模块根据按时间排序的 AIS 数据集创建轨迹。为单独处理每艘船舶，输入数据根据船舶的海上移动业务标识进行分组。海上移动业务标识基本上是唯一标识符，因为它能在特定时间唯一标识一艘船舶，但在某些情况下可能会发生变化 [1, 12]。

轨迹分为停留和移动两类，使分析可以聚焦于其中任一类别。

为构建轨迹，轨迹构建模块 会遍历按时间排序的 AIS 数据集，并逐对检查各个点。通过检查，该模块确定哪些点集代表移动中的船舶，哪些代表停留的船舶，并分别为它们创建轨迹。根据 [28] 中的定义，逐对检查还可用于改善数据质量，方法是移除时空异常值。

汇总模块 汇总模块 执行汇总查询，将轨迹数据转换为与网格相关的数据，如图 1 所示。这项工作很容易进行分布式处理和并行化，因为 Citus 会自动创建分布式查询计划。

4 数据仓库设计

DIPAAL 的设计基于一系列数据源和建议，旨在为大规模移动对象数据的分析提供高效分布式处理方案。

该设计部分基于 Kimball 和 Ross 的建议 [20]。之所以是部分基于，是因为 Kimball 和 Ross 并未针对时空和移动对象领域的某些方面提出建议，例如空间参考和表示形式。对于这些方面，该设计依据 Vaisman 和 Zimányi 的建议 [32] 以及与领域专家的咨询。总体设计采用星型模式，并以关系数据库管理系统作为物理基础，正如 [20] 中的建议。

图 3 展示了完整的数据仓库设计。该设计包含六个事实表，它们分别属于一个部分雪花化的星型模式，并具有一致性维度；时间维度和日期维度除外，它们是角色扮演维度。完整的模式是一个事实星座模式 [27]。

事实星座模式中的每个属性均由三个部分表示：最左侧部分标明该属性是否属于主键（PK）和/或外键（FK）；中间部分是属性名称；最右侧部分是数据类型。既属于 PK 又属于 FK 的属性以绿色表示，而仅属于 PK 的属性以蓝色表示。如果某个 PK 属性同时是分布属性，则以紫色渐变表示。分布属性决定行位于哪个 Citus 分片中。每个箭头均从 FK 指向其引用的属性，如图 3 所示。

指向 dim_空间_分区、dim_日期和 dim_时间的箭头未显示，但所有名称以 *di-* 开头的属性都隐含了这些箭头

愿景_标识符、日期_标识符和时间_标识符。 **时态几何点**、**时空盒**和**时态浮点数**是 MobilityDB 中的类型，其中**时态几何点**表示时空几何点，**时空盒**表示一个时空盒，而 **时态浮点数**表示时态浮点数[4]。

单元表示选用了四种粒度：5000米、1000米、200米和50米。这是与领域专家合作，并根据他们目前使用航海图的情况得出的结果。

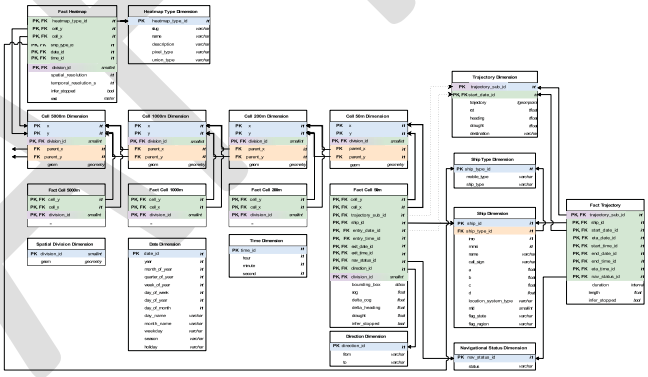


图 3：数据仓库的事实星座模式。

4.1 事实

Kimball 和 Ross 建议，事实表应以原子事件为基础构建 [20]。然而，事实表 fact_trajectory 中每一行的数据基础都是一条轨迹。fact_trajectory 使用这种粒度，因为将 AIS 点聚合为轨迹不会丢失信息；参见定义 1 和 2。

事实_轨迹包含用于描述船舶轨迹的度量，如定义 2 所述，并见图 1a。度量“持续时间”和“长度”分别表示轨迹的持续时间以及以米为单位的长度。这些都是可加度量 [20]。最后，度量 infer_stopped 是一个布尔值，true 表示该轨迹代表一艘静止的船舶，例如锚泊船舶。对于来自航行中船舶的轨迹，该值为 false。

这四种网格事实关系是基于事实_轨迹构建的聚合事实表，如图 1 所示，其中包含对已定义网格内事件的度量。这些事件描述了船舶的移动

DRAFT

情况，即船舶进入并离开网格之间的移动。对地速度是船舶在网格事件期间对地速度的平均值。对地航向变化量（COG）和航向是衡量船舶在网格内航向变化程度的度量。

事实_热力图关系包含条目，用于定义一个针对 5 × 5 千米网格的栅格。该 5 × 5 千米栅格通过存储多个像素，包含更精细的粒度。例如，分辨率为 50米 的栅格是一个包含 10,000 个像素值的 5 × 5 千米栅格。这样可以减少查询运行时间，因为需要聚合的栅格更少。

Rast 是一种高级[32] 半可加度量，因为它无法跨不同的热力图类型或粒度进行聚合。事实_热力图关系充当四种网格事实关系的中间预聚合层，关系中的条目还可以进一步聚合以生成热力图。预聚合可尽量减少查询时所需的计算，同时保留生成参数化热力图的能力。

事实_热力图关系还包含空间和时间分辨率的度量，单位分别为空间参考单位和秒。对于丹麦数据基础而言，空间参考单位是基于 EPSG:3034 的米。这些度量使系统能够存储多个分辨率。根据领域专家的要求，DIPAAL 当前预聚合了空间分辨率为四种单元粒度、时间分辨率为一天的热力图。

4.2 维度

fact_trajectory 和四个网格事实关系都引用维度 dim_trajectory。此维度负责存储轨迹的可变长度字段。它存储表示轨迹的**tgeom-point**，还存储用于表示转向率（ROT）、航向和吃水的**时态浮点数**。最后，它还包含一个**可变长字符类型**属性，用于表示轨迹的目的地。

dim_heatmap_type 存储不同热力图类型的元数据，例如如何聚合栅格，以及其名称和描述。撰写本文时，DIPAAL 会预聚合五种热力图类型：穿过网格的船舶数量、在网格中停留的累计时间、网格中航向变化量的平均值、网格中 COG 变化量的平均值，以及网格中的最小吃水。双波段栅格用于支持平均值聚合，其中一个波段表示数值总和，另一个波段表示计数。dim_heatmap_type 使得无需对数据仓库进行任何结构性更改，即可动态定义新的热力图类型，因此只需进行简单的 ETL 更改。

这四个网格维度都使用 PostGIS **geometry** 数据类型存储网格的二维索引及其物理几何。四个网格维度采用与四个单元事实相同的层次结构和粒度级别。这四个网格维度以雪花层级相互关联，较小的网格引用其较大的父网格。

采用雪花模式是为了避免数据膨胀；若将层次结构表示为反规范化星型模式层级，就会发生数据膨胀。使用 5000m 和 50米 网格对层次结构进行反规范化，会导致 5000m 粒度的数据重复 10,000 次。此外，采用雪花模式也能让今后调整层次结构变得轻而易举。

Dim_spatial_division 是一个不会变化的固定维度。它包含两个属性，分别是 id 和几何。该维度用于每一种具有空间分布的关系。第5节对空间分布方案进行了深入描述。

4.3 分布式关系和属性

维度 dim_nav_status、dim_direction、dim_spatial-_division 和 dim_time 的大小固定。dim_date 关系平均每天加载后增加一行；dim_ship 则每天按新增船舶的数量增长，但预计每天增长不会超过几百行。所有这些关系都配置为参考关系，这意味着这些关系中的所有数据都会复制到 Citus 集群的所有工作节点上，以确保数据局部性 [14]。

预计 fact_trajectory、dim_trajectory、四种单元事实以及 fact_heatmap 的增长速度最快。因此，对这些关系进行分布。四种单元事实关系和 fact_heatmap 关系使用 dim_spatial_division 维度进行空间分布，详情见第5节。四种单元维度关系与四种单元事实关系一起进行空间分布，以确保数据局部性。

要对 fact_trajectory 和 dim_trajectory 关系进行空间分布，需要将轨迹拆分为适合 dim_spatial_division 中定义的划分区域的线段。如果查询需要完整轨迹，那么拆分轨迹后就需要进行轨迹重建，这会涉及跨节点重建和重新分区。因此，决定改用随机哈希分区来分布轨迹。此设计意味着，要根据轨迹生成单元事实，就必须重新分区，因为单元关系与轨迹关系并非共置。第6.2节将对此问题进行详细讨论。

5 空间分布

DIPAAL 可以生成多种热力图。生成热力图是一项计算量很大的任务，其开销会随时间跨度和空间跨度而增加。因此，这类任务的分配方式及所采用的分布方法至关重要。

5.1 动机

为了高效地分布式计算热力图，同一像素的所有值都必须位于同一个分片中，这样便可将求和操作下推至工作节点。相同颜色的栅格表示属于同一区域的栅格，但它们可能代表不同的时间、船舶类型或移动类型。例如，两幅相同颜色的栅格中，一幅表示货船船舶类型，另一幅表示客船船舶类型。区域 指栅格覆盖的区域。

在协调节点上聚合栅格会产生高昂的网络开销，并使系统受限于协调节点的计算能力。因此，这种方式无法实现水平扩展。空间分布可确保给定区域的所有栅格都位于同一工作节点上。因此，工作节点会对每个区域执行栅格聚合，并且每个区域只向协调节点传输一幅栅格。协调节点的工作量较少；它只需将各个区域对齐并合并为一幅大型栅格。

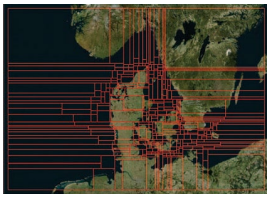
为了对空间聚合进行水平扩展，单元事实和维度与热力图在空间上共置。

DIPAAL 的空间域中基于四叉树和 kd树的空间分布如图4所示，下面将对此进行说明。

DRAFT



(a) 生成的四叉树。



(b) 生成的 kd树。

图4：四叉树和 kd树空间划分 n

5.2 空间分布方法

要扩展热力图等空间聚合，需要空间分布。下一步是确定通过划分空间域来创建空间分布的方法。

由于 DIPAAL 的最大单元粒度为 5 km，因此空间划分的边长必须是 5 km 的整数倍，划分区域也就不可能小于 5 × 5 km。

空间分布旨在创建划分区域，使各分片中的网格事实计数保持平衡。这是因为四种网格事实关系的规模远大于其他关系。创建大小相等的分片，可确保查询网格关系时的负载以及每个工作节点的存储需求均衡。由于不同粒度的网格事实数量成正比，因此采用 5000 m 网格事实的计数作为空间分布的依据。

DIPAAL 中考虑采用两种划分方法。第一种是区域四叉树方法 [31]，，第二种是 kd 树方法 [11]。R 树也可用于空间划分，但由于无法覆盖整个空间域，因此不予考虑。Li 等人 [24], 曾探索使用 R 树进行空间划分，但其无法覆盖整个空间域的特性导致需要采用变通方案，最终效果不如四叉树和 kd 树。

构造四叉树时，通过寻找能够包围数据全局空间域的最短边长来确定全局空间域，且边长应满足 s，即 $s \in 2^n * 5000m$其中 n 为自然数。边长必须是 5,000 米乘以 2 的某次幂，因为每次四叉树分割都会将网格的边长减半。这样可确保每个单元格经过 n 次划分后，边长仍能被 5 千米整除。因此，n 定义了四叉树的最大深度。每次分割时，选择单元格数量最多的划分区域，直到达到最大深度；此时若再次分割，所得划分区域将小于 5 × 5 千米。

kd 树构造采用了类似的方法。该函数的输入是全局空间域以及所创建划分区域数量的上限。

四叉树和 kd 树方法都有一个参数，用于定义所创建划分区域数量的上限。根据 2021 年的实验，选择 400 个划分区域作为最大数量。实验表明，继续增加划分区域需要将其拆分为小于 5 公里的区域，或导致划分区域之间的不平衡程度加剧。

这两种方法得出的空间划分不会随时间变化，即它们是静态的。静态特性使得可以将空间聚合下推至工作节点，而不受查询时间范围的影响。静态划分可能导致划分区域随着交通模式的演变或新安装 AIS接收器而变得不均衡。第7.4节评估了数据随时间的偏移情况。

生成的四叉树和 kd 树如图4所示，其中图4a展示四叉树方法，图4b展示 kd 树方法。

使用标准差（SD）[26] 和变异系数（CV）[6]度量数据偏差。对于这两项指标，数值越低，表示分片的均衡程度越高。2021 年数据包含 900 万条轨迹，行驶总里程达 8500 万公里，涉及 2.8 万艘不同的船舶。

在对 2021 年的 AIS 数据构建四叉树和 kd 树后，对标准差（SD）和变异系数（CV）进行度量。四叉树的标准差为 81,192，变异系数为 106%；kd 树的标准差为 31,814，变异系数为 41%。基于 kd 树的划分区域均衡性更好，因此被 DIPAAL 用作分布方法。这一结果可由数据基础的不均匀性解释：水上航行通常沿着既定海上航道行驶，因此会形成交通繁忙和稀少的区域。与较为僵化的四叉树方法相比，kd 树方法的动态特性更能捕捉这种不均匀性。因此，这些结果未必适用于其他空间域或数据集。

6 实现

本节介绍 DIPAAL 的实现细节。DIPAAL 的源代码位于 https://github.com/DIPAAL/etl。

6.1 线简化

在 ETL 流程的实现过程中，我们发现构建的轨迹中包含时空相似度较高的点，停泊的船舶尤其如此。每条轨迹都以 MobilityDB 点序列的形式存储。MobilityDB 序列会对时间戳连续的点进行线性插值，因此可以推断轨迹中任意给定时间的船舶空间位置。因此，移除对轨迹精度贡献较小的数据点，可以在不显著损失精度的情况下缩短 DIPAAL 的查询运行时间。

为提高 DIPAAL 的性能，我们对每条轨迹执行线简化[17]。在`dim_`轨迹关系中，DIPAAL 使用 10 米的误差界限进行线简化，使存储的点数减少至原来的六分之一。线简化采用 Douglas–Peucker算法[15]，并使用同步欧氏距离[29] (SED) 衡量误差，通过 MobilityDB 的`douglasPeuckerSimplify` 函数实现。SED 与欧氏距离不同，它还考虑时间维度[29]。简化过程会使轨迹产生 10 米的误差。领域专家认可引入这一误差，因为数据量减少至六分之一所带来的益处 超过了由此产生的轻微误差。

6.2 汇总

该汇总模块 负责执行汇总，从而生成轨迹数据的聚合表示形式。该模块的功能通过一组 SQL 查询实现，因为它只与数据仓库交互。

DRAFT

单元汇总 轨迹会进行聚合，以填充每种单元粒度对应的事实_单元关系。实验表明，直接将维度_单元与维度_轨迹中表示轨迹的几何进行连接，以填充事实_单元，性能较差。这是因为此类连接所需的一组 PostGIS 函数（ST_Crossing、ST_Contains 和 ST_Touches）在轨迹的最小外接矩形（MBR）较大时，索引过滤效率低下。

热力图汇总 单元事实聚合为栅格，以填充包含热力图的事实_热力图关系。

热力图的高效创建得益于 事实关系_网格 的空间分布。正如第5节所述，这种分布能够确保数据局部性。

空间分布使每个工作节点都能在本地创建整个热力图的一部分，即一个栅格。这些栅格会填充事实_热力图关系，并针对数据仓库中事实_单元的每种粒度创建。此外，每种粒度还会创建不同类型的栅格，每种栅格都聚合事实_单元中的某一类信息。

7 评估

本节评估DIPAAL上各种查询的性能，以及空间分布方法的数据倾斜
包含新旧数据。评估重点关注单用户场景下的查询运行时间，因此未对并发多用户场景进行任何评估。

7.1 硬件/软件

DIPAAL 的数据仓库集群是一个 Citus 集群，由六台配置相同的机器组成（AMD Ryzen 7 5800x、32GB 内存、2 TB 磁盘），采用一个协调节点和五个工作节点的架构。这些机器通过千兆以太网连接。

每台机器都在 Kubernetes [21] 中运行一个容器化的 PostgreSQL 15.3 实例，并安装了 PostGIS 3.3.2、MobilityDB 1.1.0 和 Citus 11.3.0 扩展。操作系统为 Ubuntu 22.04，Python 版本为 3.11。

DIPAAL 评估所使用的数据集是由 DMA 发布的公开 AIS 数据集。除非基准测试部分另有说明，否则所有基准测试均使用 2021 年的数据。

7.2 运行时间评估指标与流程

所有用于度量运行时间的评估均采用两项指标：运行时间和平均工作线程空闲比例（WIF）

运行时间是指从向数据仓库发送查询到接收到完整结果所经历的持续时间。WIF 定义为工作节点处于空闲状态的时间占运行时间的比例。较高的 WIF 表明工作节点的计算能力被浪费。WIF 按公式（1）中的定义进行计算

$$\text{WIF}_j = \frac{\max(\text{Worker Time}) - \text{Worker Time}_j}{\max(\text{Worker Time})} \quad (1)$$

在公式（1）中，工作节点时间$_j$ 是工作节点 j 完成该节点上所有分片所花费的时间，其定义见公式（2）

$$\text{Worker Time}_j = \max_{i \in \text{Worker Shards}_j} \text{Shard Time}_i \quad (2)$$

平均WIF的下限为 0%，上限为 $1 - \frac{1}{|workers|}$。上限是因为单个工作节点始终处于非空闲状态。因此，平均WIF被归一化到0%至100%的范围，如公式（3）所示

$$\text{Average WIF} = \frac{\sum WIF}{|workers|} * \frac{1}{1 - \frac{1}{|workers|}} \quad (3)$$

每个查询执行十次迭代，以从结果中剔除异常值。随后，通过删除两个最低值和两个最高值，再计算均值，使用截尾均值对指标进行汇总。

为确保结果一致，每次评估前都会重启集群中的所有 PostgreSQL 实例，并清除操作系统文件缓存
query 迭代。随后 p通过随机查询重新预热缓存。

7.3 热力图评估

本节介绍 DIPAAL 的热力图性能评估。热力图基准测试有两个目的：找出性能最佳的物理存储方案，并比较单工作节点配置与五工作节点配置，以评估扩展能力。

评估了三种物理存储方案：基于行的存储、Citus Columnar 和分区式 Citus Columnar。分区式 Citus Columnar 存储方案采用 Citus Columnar，但按热力图类型、分辨率和月份进行分区，而非使用全局月分区。分区式 Citus Columnar 方案旨在通过将可能一起查询的数据存储在同一条带中，缩短列式存储的运行时间。

在五工作节点配置上评估这三种物理存储方案。然后，在单工作节点配置上评估性能最佳的物理存储方案，以比较扩展能力。

配置的评估由一系列共 36 个查询组成。这 36 个查询是四种单元粒度、三种时间跨度和三个空间区域的笛卡尔积，见图 6。 三种时间跨度分别为 1 天（2021年2月28日）、30天（2021年1月26日至2月24日）和 1年（2021年）。

图5展示了这三个空间区域，其中奥胡斯港（46.78km^2）为红色（3），大贝尔特海峡（3 071km^2）为绿色（7），DIPAAL 的空间域（725 725km^2）为蓝色（2）。

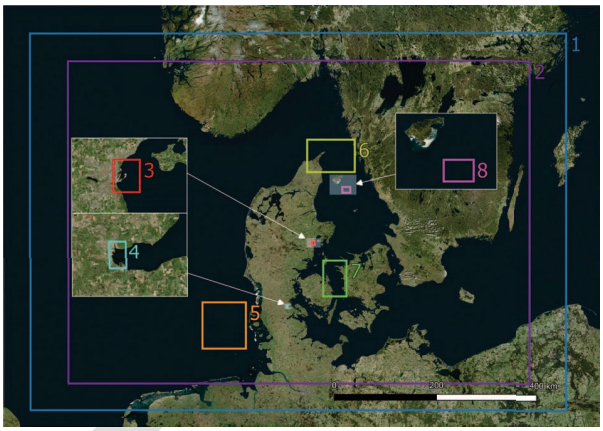


图5：八个基准区域

DRAFT

$$\left\{\begin{array}{l} Aarhus\ Harbour \\ The\ Great\ Belt \\ Spatial\ Domain\ of\ DIPAAL \end{array}\right\} \times \left\{\begin{array}{l} 1\,day \\ 1\,month \\ 1\,year \end{array}\right\} \times \left\{\begin{array}{l} 5000m \\ 1000m \\ 200m \\ 50m \end{array}\right\}$$

图6：36种热力图配置

热力图评估结果见表1。未纳入1天查询，因为它们的运行时间始终低于1秒，因此没有参考意义。

物理存储方案评估 在表1中，每项查询最快的配置以粗体标出。值得注意的是，对于大多数查询，采用基于行的存储的五工作节点配置速度最快；对于运行时间不超过1秒的查询，差异则微不足道。

由于降低慢速查询的运行时间更为重要，因此选择行式物理存储方案作为DIPAAL所使用的物理存储方案。之所以选择行式物理存储方案，是因为它在所有运行时间较长的查询中表现最佳。

扩展评估 将行式物理存储方案应用于单工作节点配置，并与五工作节点配置下的行式物理存储方案进行比较。结果见表1最右侧一列。扩展程度以百分比度量，其中100%表示运行时间相同，200%表示五工作节点配置的运行速度快一倍。

从表1可以看出，扩展情况差异显著。出现这种差异是意料之中的，因为较小的区域只涉及少数分片，这一点也体现在奥胡斯港查询较高的 WIF 上。可以看出，查询区域越大，WIF 越低，因为计算结果时涉及的分片越多。在奥胡斯港区域，涉及 5 个分片；在大贝尔特海峡区域，涉及 15 个分片；而在完整空间域中，则涉及全部 400 个分片。

对于运行时间较短的查询，即在奥胡斯港和大贝尔特海峡区域进行的查询，在采用行配置的五工作节点配置下，200米热力图的 WIF 始终低于 50米热力图。同样可以观察到，50米热力图的扩展效果不如 200米热力图。通过检查 explain analyze 转储内容，可以清楚地看出，这是由分片不平衡造成的。例如，在为期一年的大贝尔特海峡查询中，有 14 个分片参与了计算。

最大的分片包含 2.4GB 栅格数据，而下一个最大的分片包含 1.9GB 栅格数据，即大小相差 26%。通过研究查询计划和各个分片的计时结果可以观察到，栅格空间聚合的运行时间与分片中的像素总数呈线性比例关系。

7.4 空间分布偏斜

空间分布偏斜基准测试不同于其他基准测试：它评估空间分布随时间变得多么偏斜，而不是测量运行时间。

为衡量偏斜程度，采用标准差（SD）和变异系数（CV），如第5节所述；这两项指标的值越低，表示分片的平衡性越好。针对四种单元粒度，分别根据单元事实的计数和热力图栅格大小的总和来测量这些指标。

表2显示，单元事实计数和
2011年所有粒度的热力图栅格尺寸都更大
和2022年相比，2021年的尺寸较小。此外还可以看到，2022年更接近
2021年的度量值，而非2011年，这符合预期，因为相比十年间，
交通模式更可能在一年内发生变化
年份。评估结果表明，查询近年的数据可能比查询较早的数据更快。

8 相关工作

随着生成轨迹数据的设备增多，[33], 处理轨迹数据的系统必须具备可扩展性。随着数据量增加，这些系统必须实现高效的算法来查询轨迹数据。

高效处理和存储轨迹数据是一个活跃的研究领域，业界已使用多个系统 [22]。这些系统处理轨迹数据的方式各不相同。有些采用数据仓库方法 [8, 28], ，另一些则采用更通用的轨迹管理方法 [9, 22, 23, 35]。

底层存储系统各不相同，有的使用关系数据库管理系统 [28], ，有的使用 NoSQL 数据库 [22, 23], ，还有的使用基于内存存储 [35] 或分布式文件系统 [9]的处理框架。

DIPAAL 使用基于轨迹的存储模型，将完整轨迹存储为单行。其他采用或支持基于轨迹的存储模型的系统包括 Lan 等人 [22], 、Nielsen 等人 [28], 和 Li 等人 [23]。此外，还有其他存储模型，例如基于点的模型 [22, 28], ，该模型单独存储轨迹中的每个点；以及基于段的模型 [9, 22, 35], ，该模型将轨迹分段存储。DIPAAL 选择基于轨迹的模型，因为该模型在查询执行期间无需重建，且与基于点的模型 [22]相比，存储成本更低。因为无需重建，所以步骤更少。

DIPAAL 使用空间分布来缩短查询数据的运行时间，并基于底层网格单元，使用 Citus 分片来分发查询并实现并行处理。DIPAAL 选择利用一整年的数据，通过 kd 树方法创建空间划分，而类似系统则采用了其他方法，by sim- ilar systems [7, 24]。Li 等人 [24] 选择使用四叉树方法构建划分区域，因为他们要划分非点数据，而这类数据更难找到 kd划分 [24]。Aji等人 [7], 则使用一种基于阈值的自定义网格划分方法。DIPAAL 使用的空间分布方法可能会产生覆盖范围很大的划分区域。不过，与 Aji等人相比，该方法确保优先划分最大的区域，并且所有划分区域都包含数据，从而使数据在集群中分布得更加均衡。

Li 等人和 Aji等人都使用全局索引查找分片。DIPAAL 则依靠分片的本地索引，快速判断分片是否包含相关数据。

DIPAAL 的 ETL 流程会生成各种热力图。与 Liu 等人类似，DIPAAL 采用分箱方法[25] ，基于 5公里网格和空间划分生成小型热力图，再将它们组合成各种热力图。

9 结论

DIPAAL 平台由模块化 ETL管道组成，通过明确定义的清洗规则提升数据质量。ETL 流程的清洗规则和模块化设计使 DIPAAL 只需进行极少修改，即可应用于任何 AIS 数据集。DIPAAL 运行

DRAFT

			五工作线程						单工作线程	
			Row		列式		列式 分区的		Row	
			时间（秒）	WIF（%）	时间（秒）	WIF（%）	时间（秒）	WIF（%）	时间（秒）	扩展（%）
奥胡斯港	30天	200m	0.37	27	**0.35**	64	0.35	47	0.59	160
		50m	0.36	65	0.34	66	**0.34**	53	0.55	150
	1年	200m	**0.92**	56	0.94	80	1.89	35	6.37	693
		50m	1.10	68	**1.01**	82	2.83	42	4.69	424
大贝尔特海峡	30天	200m	**0.36**	21	0.47	22	0.36	19	0.93	258
		50m	**1.17**	39	1.36	37	1.28	43	1.80	154
	1年	200m	**2.19**	20	3.36	27	2.53	23	7.68	350
		50m	**12.50**	37	14.4	38	13.74	42	18.17	145
DIPAAL 的空间域	30天	200m	**1.27**	19	4.17	18	1.58	15	8.09	639
		50m	**11.90**	13	14.4	14	12.06	14	42.10	354
	1年	200m	**9.18**	9	35.58	16	9.36	12	40.19	438
		50m	**102.35**	11	126.37	10	102.81	10	内存不足	

表1：热力图基准测试结果。

		2011		2021		2022	
		SD	CV	SD	CV	SD	CV
Cell	50m	3359k	66%	272万	50%	273.4万	51%
	200m	836k	65%	671k	49%	675.6万	50%
	1000米	164k	61%	127k	43%	129k	45%
	5000米	36k	56%	32k	41%	34k	44%
Heatmap	50m	2388MB	77%	2002MB	61%	1997MB	61%
	200m	164MB	73%	136MB	56%	137MB	57%
	1000米	18MB	56%	14MB	40%	15MB	42%
	5000米	11MB	52%	9MB	38%	9MB	40%

表2：空间分布偏斜度

在由六台商用机器组成的集群上运行，ETL 流程中的数据准备部分能够独立运行。

对空间分布的评估表明，静态划分区域会导致分片之间的负载平衡随着时间推移而变得不均。

致谢

本文获得了欧盟资助的 MobiSpaces 项目的资助，资助协议编号为 101070279。

参考文献

[1] [无日期。]海上移动业务中身份标识的分配与使用。https: // www.itu.int/rec/R-REC-M.585-9-202205-I/en 访问日期：2023/06/23。[2] 2023可能与第二次爆炸同时发生的第三次爆炸。https://eng.geus.dk/ about/news/news-archive/2022/september/newspage 访问日期：2023/02/06。[3] 2024Citus Data。https://www.citusdata.com/ 访问日期：2024/02/12。[4] 2024 MobilityDB。https://www.mobilitydb.com/ 访问日期：2024/02/12。[5] 2024 PostGIS。https://postgis.net/ 访问日期：2024/02/12。[6] H. Abdi。2010年。变异系数。《研究设计百科全书》第1卷，第5期（2010年）。[7] A. Aji、F. Wang、H. Vo、R. Lee、Q. Liu、X. Zhang 和 J. Saltz。2013年。Hadoop GIS：基于 MapReduce 的高性能空间数据仓库系统。《VLDB Endowment 会议论文集》第6卷，第11期（2013年），1009–1020。[8] T. Alsahfi、M. Almotairi 和 R. Elmasri。2020年。轨迹数据仓库综述。《空间信息研究》第28卷，第1期（2020年），53–66。[9] M. Bakli、M. Sakr 和 T. H. A. Soliman。2019年。HadoopTrajectory：Hadoop 时空数据处理扩展。《地理系统杂志》第21卷，第2期（2019年），211–235。[10] M. Bakli、M. Sakr 和 E. Zimanyi。2019年。MobilityDB 中的分布式移动对象数据管理。载于《BigSpatial ’19》。[11] J. L. Bentley。1975年。用于关联搜索的多维二叉搜索树。《ACM 通讯》第18卷，第9期（1975年9月），509–517。

[12] K. Bereta、K. Chatzikokolakis 和 D. Zissis。2021年。海事报告系统。施普林格国际出版公司，Cham，3–30。 [13] U. Cubukcu、O. Erdogan、S. Pathak、S. Sannakkayala 和 M. Slot。2021年。Citus：面向数据密集型应用的分布式 PostgreSQL。收录于*SIGMOD*。2490–2502。 [14] Citus Data。2022年。Citus 文档 — Citus 11.1 文档。https: //docs.citusdata.com/_/downloads/en/v11.1/pdf/ 访问日期：2022年12月15日。 [15] D. H. Douglas 和 T. K. Peucker。1973年。用于减少表示数字化线条或其示意图所需点数的算法。*Cartographica* 10 (1973)，112–122。 [16] T. Emmens、C. Amrit、A. Abdi 和 M. Ghosh。2021年。自动识别系统数据的前景与风险。《应用专家系统》178 (2021)，114975。 [17] M. Fleischmann。2020年。折线简化算法。访问日期：2022年12月16日。

[18] A. Graser。2021年。用于识别连续移动数据中问题的探索性数据分析协议。《 LBS 杂志》第15卷，第2期（2021年），89–117。[19] PostgreSQL 全球开发组。2022年。PostgreSQL 15.1 文档。https://www.postgresql.org/docs/15/index.html 访问日期：2022/12/19。[20] R. Kimball 和 M. Ross。2013年。《数据仓库工具箱：维度建模权威指南》（第3版）。Wiley Publishing。[21] Kubernetes。2023年。https://kubernetes.io/docs/home/ 访问日期：2023/06/23。[22] H. Lan、J. Xie、Z. Bao、F. Li、W. Tian、F. Wang、S. Wang 和 A. Zhang。2022年。VRE：一种多功能、稳健且经济的轨迹数据系统。《VLDB Endowment 会议论文集》第15卷，第12期（2022年9月），3398–3410。[23] R. Li、H. He、R. Wang、S. Ruan、Y. Sui、J. Bao 和 Y. Zheng。2020 年。TrajMesa：用于大规模轨迹数据的分布式 NoSQL 存储引擎。载于《ICDE》，2002–2005。[24] R. Li、R. Wang、J. Liu、Z. Yu、H. He、T. He、S. Ruan、J. Bao、C. Chen、F. Gu 等。2021年。分布式时空 k 近邻连接。载于《SIGSPATIAL》，435–445。[25] Z. Liu、B. Jiang 和 J. Heer。2013年。imMens：大数据实时可视化查询。《计算机图形论坛》第32卷，第3pt4期（2013年）。[26] S. H. Khan M.。2011年。标准差。Springer Berlin Heidelberg，柏林、海德堡，

1378 1379 10 1007 978 3 642 04898 2_535 –。

[27] 2008

https://doi.org/ 。 / - - - - D. Mishra、A. Yazici 和 B. Pinar Basaran。 。数据仓库中数据模型的案例研究。载于《ICADIWT》，314–319。[28] K. S. H. Nielsen、M. E. Olsen、P. K. Egede 和 A. Øztürk。2022年。在多用途 AIS 数据仓库中分析数十亿条 AIS 记录。硕士论文。AAU。[29] M. Potamias、K. Patroumpas 和 T. Sellis。2006年。基于时空标准对轨迹流进行采样。载于《SSDBM》，275–284。[30] P. J Sadalage 和 M. Fowler。2013年。《NoSQL 精要：新兴多语言持久化世界简明指南》。Pearson Education。[31] H. Samet。1988年。四叉树、八叉树及相关层次数据结构概述。《计算机图形学与 CAD 的理论基础》（1988年），51–68。[32] A. Vaisman 和 E. Zimányi。2022年。空间与移动数据仓库。Springer Berlin Heidelberg，柏林、海德堡，437–486。[33] S. Wang、Z. Bao、J. S. Culpepper 和 G. Cong。2021年。轨迹数据管理、分析与学习综述。《ACM 计算综述》第54卷，第2期（2021年）。[34] Zhe Xiao、Xiuju Fu、Liye Zhang 和 Rick Siow Mong Goh。2020年。海上交通服务网络中的交通模式挖掘与预测技术：全面综述。《IEEE 智能交通系统》第21卷，第5期（2020年），1796–1825。[35] Z. Zhang、C. Jin、J. Mao、X. Yang 和 A. Zhou。2017年。TrajSpark：面向大规模轨迹数据的可扩展高效内存管理系统。载于《Web 与大数据》，11–26。

DRAFT