# A Survey on Spatio-Temporal Big Data Analytics Ecosystem: Resource Management, Processing Platform, and Applications

Huanghuang Liang, Zheng Zhang, Chuang Hu, *Member, IEEE*, Yili Gong, *Member, IEEE*, and Dazhao Cheng, *Senior Member, IEEE*

*(Survey Paper)*

***Abstract*—With the rapid evolution of the Internet, Internet of Things (IoT), and geographic information systems (GIS), spatio-temporal Big Data (STBD) is experiencing exponential growth, marking the onset of the STBD era. Recent studies have concentrated on developing algorithms and techniques for the collection, management, storage, processing, analysis, and visualization of STBD. Researchers have made significant advancements by enhancing STBD handling techniques, creating novel systems, and integrating spatio-temporal support into existing systems. However, these studies often neglect resource management and system optimization, crucial factors for enhancing the efficiency of STBD processing and applications. Additionally, the transition of STBD to the innovative Cloud-Edge-End unified computing system needs to be noticed. In this survey, we comprehensively explore the entire ecosystem of STBD analytics systems. We delineate the STBD analytics ecosystem and categorize the technologies used to process GIS data into five modules: STBD, computation resources, processing platform, resource management, and applications. Specifically, we subdivide STBD and its applications into geoscience-oriented and human-social activity-oriented. Within the processing platform module, we further categorize it into the data management layer (DBMS-GIS), data processing layer (BigData-GIS), data analysis layer (AI-GIS), and cloud native layer (Cloud-GIS). The resource management module and each layer in the processing platform are classified into three categories: task-oriented, resource-oriented, and cloud-based. Finally, we propose research agendas for potential future developments.**

***Index Terms*—Artificial intelligence framework, Big Data system, cloud platform, database management system, geographic information system, resource management, spatio-temporal Big Data.**

Manuscript received 28 February 2023; revised 13 October 2023; accepted 15 November 2023. Date of publication 13 December 2023; date of current version 14 March 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFE0205700, in part by the National Natural Science Foundation of China under Grant 62302348, in part by the Fundamental Research Funds for the Central Universities under Grant 2042023kf0132, in part by the General Program of Hubei Provincial Natural Science Foundation of Chinan under Grant 2023AFB831, and in part by the Special Fund of Hubei Luojia Laboratory under Grant 220100016. Recommended for acceptance by M. Huang. *(Corresponding authors: Yili Gong; Dazhao Cheng.)*

The authors are with the School of Computer Science, Wuhan University, Wuhan, Hubei 430072, China (e-mail: hhliang@whu.edu.cn; zzhang3031@whu.edu.cn; handc@whu.edu.cn; yiligong@whu.edu.cn; dcheng@whu.edu.cn).

Digital Object Identifier 10.1109/TBDATA.2023.3342619

## I. Introduction

With the rapid expansion of the Internet, IoT, GIS, and data collection technologies becoming more diverse and data types continually enhancing, making **STBD** grows "explosively." Integrating spatio-temporal information and Big Data marks the official entry into the era of STBD [1]. STBD compensates for the lack of data by providing a rich volume and variety of data types that can fully satisfy numerous research needs, promoting ongoing, in-depth, cross-sectional research. The dynamic evolution of spatio-temporal objects, events, and other elements and their dynamic correlations pose significant challenges to data management, processing, and analysis [2]. Meanwhile, computer technology is rapidly evolving. Computing power has skyrocketed, and hardware architecture has evolved. Big data processing frameworks and high-performance computing models have been established. Due to the evolution of new hardware, software, and application requirements, expanding GIS to improve adaptability and performance efficiency presents new opportunities and challenges.

In the STBD ecosystem, GIS data merges spatial and temporal dimensions, capturing changes in geographical information. GIS consolidates varied data into a unified spatial framework, enabling in-depth spatial-temporal analysis [3]. Its unique attributes allow for integrating different data types and unveiling hidden patterns within geographical spaces, enhancing our understanding of spatial trends and contexts. The seamless integration of databases, Big Data, AI, and cloud computing signifies an emerging trend in unified systems [4]. Database systems are foundational for managing vast amounts of structured and unstructured data. Big data and AI shine in sophisticated data analytics, whereas cloud platforms amplify processing efficiency through resource provisioning. STBD, with its expansive, varied, and time-sensitive attributes, is deeply linked with these technologies. Databases paired with Big Data analytics manage and decipher extensive datasets, revealing pivotal insights. AI, fueled by data-driven learning, infuses intelligence into processing and decision-making. Cloud infrastructure offers unmatched scalability, adapting to diverse processing needs [5]. Their collective synergy in a cohesive ecosystem amplifies combined strengths, bolstering data processing capabilities.

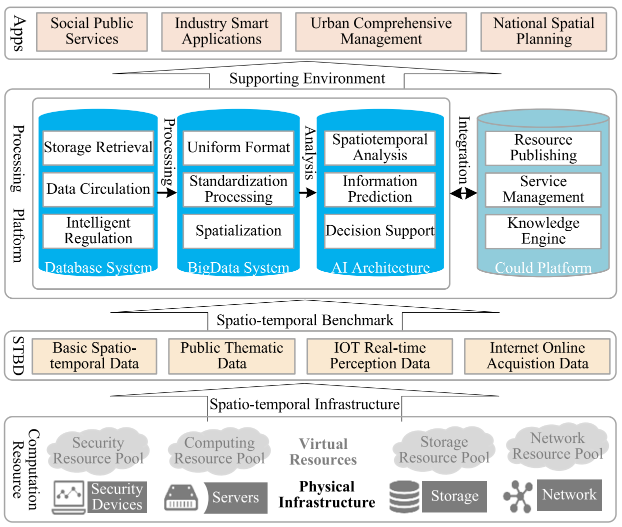

Fig. 1. Applications of smart city STBD platform.

Spatio-temporal applications typically span multiple domains, including GIS, AI, and Big Data. Given the large-scale nature of spatio-temporal data, there is an urgent need for robust data processing capabilities to ensure efficient analysis and management. spatio-temporal applications involve intricate data processing involving geographic location and time, endowing them with unique features such as time series and geographical coordinates [6]. In spatio-temporal applications, achieving real-time and accurate data processing is crucial, especially in traffic monitoring and weather forecasting applications. This underscores the demand for timely data updates and precise spatial information. So, it's critical to integrate these areas to meet interdisciplinary needs. In healthcare [7], this convergence empowers predictive analytics for patient outcomes, while in finance [8], it facilitates fraud detection and risk assessment. We have provided application examples for the smart city STBD platform [9], underscoring the interconnectedness of computational resources, spatio-temporal data, processing platforms, and applications in Fig. 1. Collectively, this convergence represents a multifaceted synergy that transcends technology silos, unleashing transformative potential and opportunities across diverse sectors.

Resource management is indispensable for STBD processing efficiency across various platforms. In DBMS, meticulous resource allocation ensures efficient storage and retrieval of STBD, providing robust support for complex queries and analyses. Intelligent resource allocation in Big Data processing systems significantly enhances the processing speed of massive datasets, accelerating information extraction and pattern recognition. In AI training architectures, precise resource management directly impacts the performance of machine learning (ML) models by optimizing computational resources to improve training efficiency. Dynamic resource allocation and management on cloud platforms are crucial for ensuring system flexibility and adaptability, especially in addressing evolving demands for data processing and AI training.

To better understand resource management's role in the multi-platform environment of STBD processing, we sort out the existing STBD analytics ecosystem and divide it into the five modules in Fig. 2. (1) **STBD** included definition, characteristics, and analysis. (2) **Computation resources** comprised computational, storage, and communication resources used in distributed computing architecture to perform various tasks. (3) **Processing Platform** contained data management, processing, analytics, and cloud native layers. (4) **Resource Management** covered DBMS-GIS, BigData-GIS, AI-GIS, and Cloud-GIS resource scheduling and management policies. (5) **Applications** of STBD oriented to geoscience and human-social activities. Despite their differences in nature and objectives, these layers intersect and provide comprehensive solutions for addressing complex data challenges in data processing.

We have identified critical gaps that demand urgent attention. Current investigations and performance analyses are predominantly confined to systems developed before 2017, necessitating more recent research to provide additional insights. Current research predominantly focuses on developing algorithms and technologies for capturing, storing, managing, analyzing, and visualizing STBD [10], [11]. However, these efforts often need to pay more attention to the critical aspects of system optimization and resource management, limiting the potential for enhancing the efficiency of STBD processing and applications. Additionally, there needs to be more research on broader aspects of the spatio-temporal analysis ecosystem, including spatio-temporal database management, parallel processing, AI analytics, and cloud-based spatial feature integration. Research on the migration of STBD to new Cloud-Edge-End integrated computing systems is still in its infancy and represents a critical area that requires in-depth exploration [12]. These key domains demand attention in current research to fill significant gaps in understanding STBD and its practical applications. We comprehensively review recent research on STBD systems. To furnish users with a guide to address problems and employ solution techniques related to STBD. We systematically organize the state-of-the-art STBD analytics systems and delve into their resource management, providing users with a detailed understanding of the current landscape. Furthermore, we illuminate trends, emerging requirements, and challenges in each support layer within the STBD ecosystem. This offers researchers valuable insights into potential directions for future research in STBD processing and applications. Also, we'd like to explore the existing challenges and forthcoming opportunities in STBD analytics, which is our vision for the future development of STBD analytics.

The paper is structured as follows: Section II introduces the STBD's definition, characteristics, and analysis. Section III focuses on the core and spatio-temporal platforms within the data management and processing layers, with insights into spatio-temporal cloud platforms. Section IV discusses resource management strategies across these layers. Section V reviews open-source databases and STBD-related applications. Section VI highlights DBMS-GIS, BigData-GIS, AI-GIS, and cloud-GIS research trends. Section VII concludes.

## II. Spatio-Temporal Big Data

### A. Definition

STBD is an extensive, large-scale dataset founded on a unified spatio-temporal datum (comprising temporal reference system

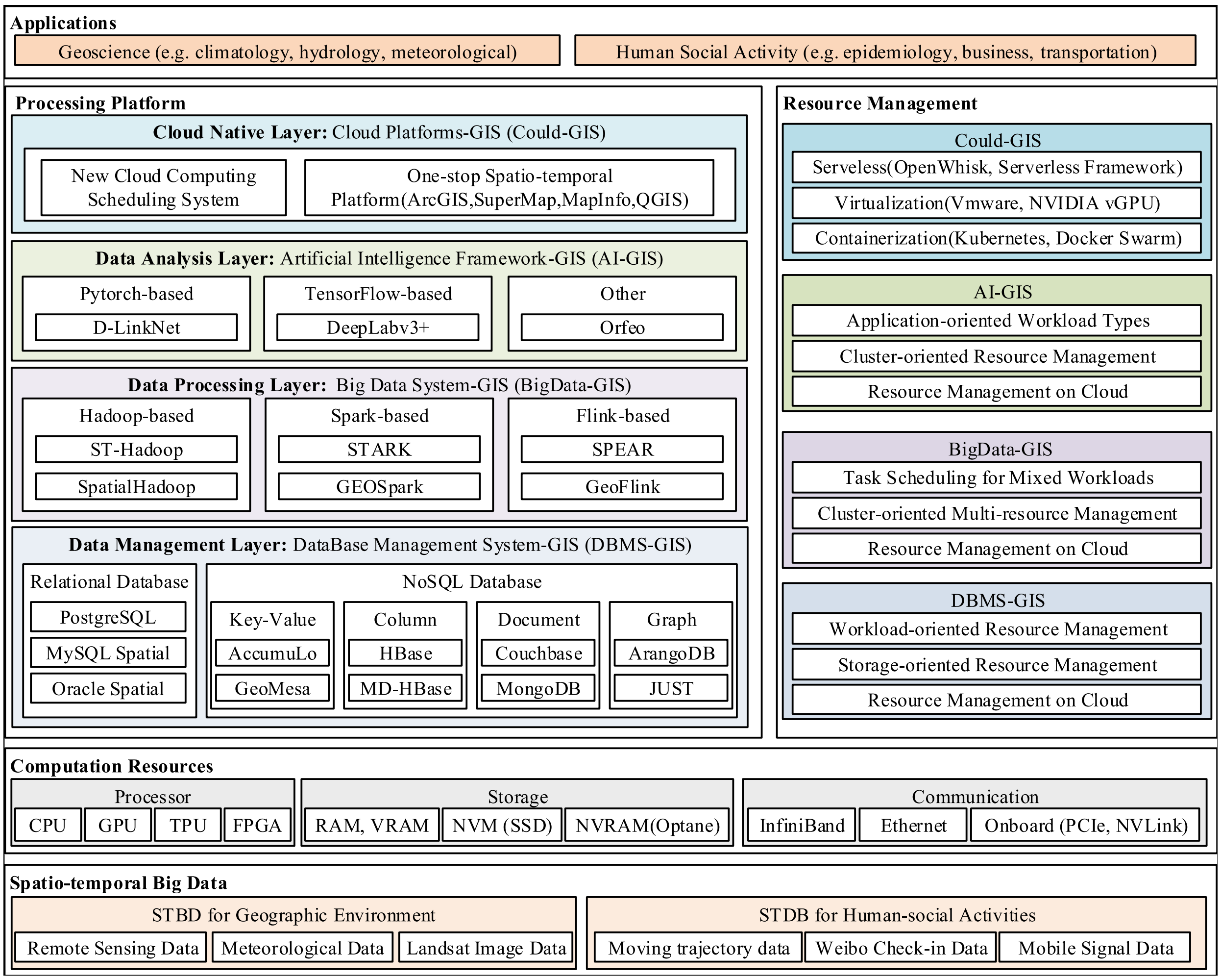
Applications
Geoscience (e.g. climatology, hydrology, meteorological)
Human Social Activity (e.g. epidemiology, business, transportation)
Processing Platform
Cloud Native Layer: Cloud Platforms-GIS (Could-GIS)
New Cloud Computing Scheduling System
One-stop Spatio-temporal Platform(ArcGIS,SuperMap,MapInfo,QGIS)
Data Analysis Layer: Artificial Intelligence Framework-GIS (AI-GIS)
Pytorch-based
D-LinkNet
TensorFlow-based
DeepLabv3+
Other
Orfeo
Data Processing Layer: Big Data System-GIS (BigData-GIS)
Hadoop-based
ST-Hadoop
SpatialHadoop
Spark-based
STARK
GEOSpark
Flink-based
SPEAR
GeoFlink
Data Management Layer: DataBase Management System-GIS (DBMS-GIS)
Relational Database
PostgreSQL
MySQL Spatial
Oracle Spatial
NoSQL Database
Key-Value
AccumuLo
GeoMesa
Column
HBase
MD-HBase
Document
Couchbase
MongoDB
Graph
ArangoDB
JUST
Resource Management
Could-GIS
Serveless(OpenWhisk, Serverless Framework)
Virtualization(Vmware, NVIDIA vGPU)
Containerization(Kubernetes, Docker Swarm)
AI-GIS
Application-oriented Workload Types
Cluster-oriented Resource Management
Resource Management on Cloud
BigData-GIS
Task Scheduling for Mixed Workloads
Cluster-oriented Multi-resource Management
Resource Management on Cloud
DBMS-GIS
Workload-oriented Resource Management
Storage-oriented Resource Management
Resource Management on Cloud
Computation Resources
Processor
CPU
GPU
TPU
FPGA
Storage
RAM, VRAM
NVM (SSD)
NVRAM(Optane)
Communication
InfiniBand
Ethernet
Onboard (PCIe, NVLink)
Spatio-temporal Big Data
STBD for Geographic Environment
Remote Sensing Data
Meteorological Data
Landsat Image Data
STDB for Human-social Activities
Moving trajectory data
Weibo Check-in Data
Mobile Signal Data

Fig. 2. Spatio-temporal Big Data analytics ecosystem.

and spatial reference system), capturing activities (such as movement changes) in both time and space, directly associated with a location through positioning or indirectly through spatial distribution [13]. Traditionally, STBD has found applications in describing meteorological data derived from sophisticated systems like remote sensing (RS), GIS, global positioning systems (GPS), geological information systems, intelligent city systems, traffic information systems, and environmental information systems. RS generally denotes non-contact, long-distance comprehensive detection technologies; GIS is a computer system for collecting, storing, managing, processing, retrieving, analyzing, and expressing geospatial data; and GPS is a high-precision radio navigation system that relies on artificial earth satellites and time information. With the incorporation of the temporal dimension, GIS has transformed into temporal GIS, representing an evolving capacity to integrate temporal data seamlessly with location and attribute data.

## B. Characteristics

STBD integrates Big Data and spatial data characteristics, incorporating time-dimension information to address various time-related geographic data challenges effectively. Big data processing involves leveraging all available data without compromising the time required for random analysis.

Big data has five key features: volume, velocity, variety, value, and veracity [14]. Spatial data can be represented in two formats: raster and vector. Raster data, exemplified by satellite imagery, is typically displayed as multi-dimensional arrays. In contrast, vector data, consisting of points, lines, and polygons, depicts geographic features like roads, regional boundaries, and GPS coordinates. Notably, STBD is distinguished by specificity, fuzziness, dynamism, finiteness, social network, heterogeneity, low quality, non-smoothness, and networking [15].

STBD's data content encompasses time, spatial, and non-spatial-temporal attributes [16]. Time attributes describe temporal information, such as time stamps for spatial objects, raster layers, or a series of snapshots. Spatial attributes encompass spatial information, including location (e.g., longitude and latitude), extent (e.g., area and perimeter), and shape composition. Non-spatio-temporal attributes, ranging from structured data like climate or demographic data to unstructured data like remote sensing images, add diversity to the dataset. The temporal snapshots model timestamps spatial layers with a shared theme, portraying lines, polygons, raster time series trajectories, and dynamic spatio-temporal networks like time-expanded or

TABLE I
SPATIO-TEMPORAL BIG DATA CHARACTERISTIC DIVISION

| Spatio data | Temporal snapshots (Time series) | Temporal change (Delta/Derivative) | Events or processes |
|---|---|---|---|
| Object | Trajectories, Spatial time series | Motion, Speed, Split or merge, Traffic model | Spatio or spatio-temporal point process, Human behavior |
| Field | Raster time series | Change across raster snapshots | Cellular automation |
| Network | Spatio-temporal network, Traffic flow | Addition or removal of nodes, Edges | Epidemiological |

time-aggregated graphs. Spatio-temporal networks depict dynamic structures across time and space, illustrating connections such as traffic routes and communication networks. Traffic flow involves the movement of entities at specific instances and places, necessitating the analysis and modeling of spatio-temporal data. The temporal change model captures spatio-temporal data, starting with an initial spatial layer at a specific time, followed by incremental modifications, covering motion (e.g., Brownian motion or random walks), velocity, and acceleration at spatial points. Splitting or merging signifies the dividing or amalgamation of lines and polygons at a specific spatio-temporal nexus. Conversely, the traffic model focuses on temporal shifts in traffic attributes, including flow, speed, and direction from a specified spatial origin. Event and process models articulate temporal details through events or processes in Table I. Given the diversity of these data types, effective analysis and processing necessitate using different methods.

## C. Analysis

*Spatial Analysis:* The continuous advancements in GIS and network technology have markedly augmented the capabilities of GIS spatial analysis. Spatial analysis, a basic function of GIS, enables the examination of spatial layout, aggregation level, and coupling association [17]. Researchers have expanded spatial analysis functions to encompass dynamic content, including geometric analysis, terrain analysis, network analysis, raster data analysis, and spatial statistical analysis. This expansion enhances the feasibility of diverse spatial planning and design decisions by bolstering data support and design science in system designs [18].

*Spatio-Temporal Analysis:* Traditionally, spatial statistics faced limitations due to the constrained accessibility of STBD [19]. However, recent advancements in social and environmental perception have significantly enhanced STBD accessibility, fueling the demand for applications and research requiring real-time and dynamic analysis of extensive spatial data [20]. This shift has prompted the development of spatio-temporal analysis methods, extending the reach of conventional spatial statistical methods into the spatio-temporal domain [21]. To fully exploit STBD, researchers must delve into the spatio-temporal analysis of environmental and human factors and their intricate interconnections. The advancement of STBD necessitates urgent theoretical, technical, and methodological support for spatio-temporal cluster/anomaly/correlation/prediction analysis.

## III. PROCESSING PLATFORM

In this section, from the data management layer, where the marriage of database management systems (DBMS) and GIS unfolds, to the dynamic realms of data processing with the infusion of the data processing layer, the data analysis layer introduces the transformative influence of AI, reshaping how we glean insights from spatial data. Finally, ascending to the cloud native layer, the fusion of technology and geography manifests in Cloud-GIS, offering scalability and adaptability that transcend traditional boundaries.

TABLE II
BASIC DATABASE MANAGEMENT SYSTEM

| Analysis tool | | PostGIS | MySQL Spatial | MongoDB | Neo4j Spatial |
|---|---|---|---|---|---|
| Data model | | Relation | Relation | Document | Graph |
| Data formats | WKT | Y | Y | N | Y |
| | GML | Y | N | N | N |
| | GeoJSON | Y | N | Y | N |
| | SVG | Y | N | N | N |
| Spatial functions | Disjoint | Y | Y | N | Y |
| | Transform | Y | N | Y | Y |
| | Aggregation | Y | N | Y | N |
| | $k$NN | Y | N | N | N |
| SQL-like query | | Y | Y | N | Cypher |

## A. Data Management Layer

The data management layer encompasses the classification of spatio-temporal databases, recognizing the limitations of traditional databases in efficiently handling massive, diverse, unstructured administrative data and heterogeneous memory resources. The demand for a usable, scalable, update-supporting spatio-temporal DBMS is evident for the successful support of contemporary urban applications.

*1) Basic DBMS:* Spatial DBMSs [22] fall into two primary categories: relational and NoSQL DBMSs. Relational DBMSs comprise interconnected two-dimensional row tables and utilize SQL for data manipulation. Well-known relational DBMSs with spatial capabilities include Oracle Spatial, IBM Db2, Microsoft SQL Server, Microsoft Access, MySQL Spatial, and PostGIS in Table II. While traditional relational DBMSs are reliable, mature, and efficient and find widespread use in various applications, they may face challenges when dealing with large-scale data, diverse data types, and the demands of emerging ultra-large-scale and high-concurrency web 2.0 platforms. Conversely, NoSQL DBMSs offer high scalability, performance, flexible data models, and robust availability. They encompass three main data store types: column-based databases (e.g., HBase), document stores (e.g., MongoDB), and graph databases (e.g., Neo4j), aptly addressing the complexities of modern applications.

The widespread adoption of in-memory databases, including Redis, NuoDB, and MySQL Cluster, has triggered a resurgence in in-memory computing technologies. However, to achieve significant performance improvements in near-memory computing, the system design must be customized to the specific requirements of the upper-layer application. Pregel [23] proposed various near-memory computing architectures tailored for MapReduce workloads. Q-PIM [24] introduced a flexible

TABLE III
NoSQL-Based Spatio-Temporal Database Management System

| Name | System Type | Base system | Data types | Partitioning | Supported queries |
|---|---|---|---|---|---|
| MD-HBase [30] | Spatio-temporal | HBase | Point, Timestamp | Range-Partition | Range, kNN |
| GeoMesa [31] | Spatio-temporal | Accumulo | Point, LineString, Timestamp, Polygon | Spatial, Temporal, Attribute | Range |
| Distributed SECONDO [32] | Spatio-temporal | Cassandra | Point, LineString, Region, Instant, Period, Periods, Interval | 3D Grid | Join |
| BBoxDB [33] | Spatial | Key-BBox-Value store | Point, LineString, Polygon | Grid, KD-Tree, Quad-Tree | Join |
| THBase [34] | Spatio-temporal | HBase | Point, Timestamp | MO-based Model | Single-Object, Range, $k$NN |
| JUST [28] | Spatio-temporal | HBase | Geom, Timestamp, Spatio-temporal_Series, T_Series | N/A | Range, $k$NN |
| TrajMesa [29] | Spatio-temporal | GeoMesa | Point, Timestamp | N/A | ID-Temporal, $k$NN, Range, Similarity |

SRAM-based precision all-digital in-memory (PIM) architecture. It utilizes a genetic algorithm-based training-free layer quantization approach to optimally control the precision of each DNN layer for enhanced efficiency. Near-memory computing still needs more efficient and transparent system-level support despite these advancements. Addressing this gap, Vermij et al. [25] put forth a dynamic workload balancing technique. This approach empowers applications to execute heterogeneously on near-memory CPUs, optimizing CPU utilization and enhancing overall performance, especially when dealing with massive datasets.

*2) Spatio-Temporal DBMS-GIS:* While traditional relational databases such as Oracle Spatial, MySQL Spatial, and PostGIS offer support for managing STBD, they often encounter challenges with scalability as data volumes grow [26]. On the other hand, distributed NoSQL data stores exhibit impressive capabilities for handling millions of updates per second. However, these NoSQL solutions inherently need more support for STBD management, primarily due to the absence of essential secondary indexes [27]. Several robust STBD analytics systems have emerged recently, capitalizing on NoSQL DBMSs like MD-HBase, GeoMesa, Distributed SECONDO, BBoxDB, THBase, JUST, and TrajMesa. Table III presents a detailed characteristic matrix of these systems. For instance, JUST [28] leverages HBase as its foundational storage, GeoMesa as its STBD indexing tool, and Spark as its execution engine. In contrast, TrajMesa [29] stands out as the first endeavor to establish a comprehensive distributed NoSQL trajectory storage engine based on GeoMesa.

## B. Data Processing Layer

The data processing layer is a critical component that houses diverse spatio-temporal platforms tailored for Big Data computing. Using Big Data systems to construct clusters can expedite the processing of terabyte (TB) or exabyte (EB)-level data [35]. Nevertheless, most Big Data systems only support traditional relational data or some STBDs. With the rapid development of location-based services and RS, many companies and projects are looking to integrate Big Data systems and STBDs.

*1) Basic Big Data System:* This section compares the three most popular Big Data systems, Apache Hadoop, Spark, and Flink in Table IV.

Apache Hadoop [36] is an open-source software framework developed by Yahoo! for distributed storage and processing large datasets with cluster-level fault tolerance. It can be set up across a cluster of computers constructed from commodity hardware. Currently, Hadoop consists of five core modules: (1) Hadoop Common is a set of shared programming libraries the other

TABLE IV
Characteristic Comparison of Apache Hadoop, Spark, and Flink

| Platform | Hadoop | Spark | Flink |
|---|---|---|---|
| Processing Engine | Batch Processing | Real Time, Micro Batch Streaming | Run time streaming, Kappa or Lambda |
| Storage Layer | HDFS, YARN, MapReduce | MapReduce | MapReduce, Storm |
| In-memory Processing | N | DAG | Explicit memory management |
| Processing Power | Petabytes data | 100x faster | 1000s nodes |
| Fault Tolerance | Clusters | High | Exactly-once processing |
| Scalability | Distributed/Grid | Parallel/Cluster | Distributed |
| Reliability | Single batch | Objects/Data Stream | Objects/Data Stream |
| Programming Language | Java | Java/Python/R/Scala | Java/Scala |

modules utilize. (2) Hadoop Distributed File System (HDFS) is a Java-based file system for storing data across multiple machines. (3) Hadoop YARN manages and schedules resource requests in a distributed environment. (4) Hadoop MapReduce is a programming model for processing large datasets in parallel. (5) Hadoop Ozone, which is an object store for Hadoop. Landset et al. [37] surveyed Hadoop's static and dynamic resource management modes, job scheduling methods, and performance comparison. However, due to its disk-based data processing, low-level processing API, and Java-only support, Hadoop is unsuitable for massive-scale and complex Big Data processing.

Apache Spark [38] supports in-memory computing and acyclic data flow through its DAG execution engine. It has been reported to be 100 times faster than Hadoop's MapReduce. Currently, research on Spark resource allocation mainly focuses on increasing its scheduling depth (e.g., heuristic attribute reduction and eviction mechanisms) and breadth (e.g., cross-platform), heterogeneous resources). Leveraging SQL, ML, graph computing, and multiple languages, Spark proves ideal for applications requiring high throughput, handling limited real-time and micro-batch data processing with substantial data volumes and intricate logic. In contrast to Hadoop MapReduce, Spark introduces a novel programming paradigm focused on the resilient distributed dataset (RDD). This dataset can be efficiently stored in memory and distributed across machines for streamlined data processing.

Apache Flink [39] is a unified stream and batch processing framework utilizing a dataflow programming model to provide event-at-a-time processing on finite and infinite datasets. This primary processing mode gives Flink lower stream processing latency than Spark Streaming, making it suitable for tasks requiring low latency, such as real-time monitoring, reports, stream data analysis, and data warehousing. Flink also supports simple static resource management, dynamic management of computing resources, and memory resource isolation, though it does not support CPU resource isolation. It can integrate with Yarn, Mesos, K8s, and other frameworks better [40].

*2) Spatio-Temporal BigData-GIS:* In Big Data processing, prevalent models include distributed computing frameworks

TABLE V
CHARACTERISTIC COMPARISON OF HADOOP/SPARK/FLINK-BASED BIGDATA-GIS SYSTEMS

| Platforms | | Hadoop-based | | Spark-based | | Flink-based | |
|---|---|---|---|---|---|---|---|
| | | Spatial-Hadoop | ST-Hadoop | GeoSpark | STARK | GeoFlink | SPEAR |
| Scalability | | Y | Y | Limited | Limited | Y | Y |
| Query Language | | Pigeon | Extended Pigeon | Extended Spark SQL | Piglet | N/A | N/A |
| System Type | | Spatial | ST | Spatial | ST | Spatial | ST |
| Data Type | | Point, Line String, Polygon | STPoint, Time, Interval | Point, Line String, Polygon, Rectangle | STObject (geo, time) | Point, Line String, Polygon, Streaming | Point, Line, Polygon, Streaming |
| Trajectory | | Slower than ST-Hadoop | Low performance | Low performance | Low performance | Fast | Fast |
| Streaming Workloads | | N | N | N | Extensions | Y | Y |
| Partitioning | | Fixed-Grid, STR | Time-Slice, Data-Slice | Uniform-Grid Voronoi, R/Quad/KDB-Tree | Fixed-Grid, Binary-Space | Grid | Dynamic-Grid Based on GeoHash |
| Index Type | | Two-Level: Grid, R-tree, R+-tree | Two-Level: Temporal, Spatial | R-Tree, Quad-Tree | R-Tree (Live & Persistent) | Grid-based | Two-Level: Spatail, Temporal |
| Supported Queries | Joins | Y | Y | Y | Y | Y | Y |
| | Range | Y | Y | Y | N | Y | Y |
| | kNN | Y | N | Y | Y | Y | Y |

built upon DAGs or massively parallel processing iterative computing models. The dynamic task generation at runtime challenges pre-scheduling approaches [41]. Given the distributed nature of these systems, there is a pronounced need for a flexible and adaptable scheduling system to effectively manage the intricate computational landscape. The goal of integrating STBD and Big Data systems is to enable traditional Big Data systems to support the storage of basic STBD types such as points, lines, planes, and time series, and the operations and analysis of STBD such as joins, range, and $k$NN. We classified into three groups: (1) Hadoop-based, (2) Spark-based, and (3) Flink-based in Table V.

*Hadoop-Based Big Data Systems* are the foundation of STBD framework research. SpatialHadoop [42] is a Hadoop MapReduce framework specifically designed to facilitate spatial data storage, processing, and querying. This framework consists of four layers: a two-level spatial index for data storage, MapReduce for data processing, basic spatial operations (e.g., range query, $k$NN, spatial join), and Pigeon-based spatial queries. These components make SpatialHadoop both convenient and extensible. Moreover, ST-Hadoop [43] provides built-in STBD types and operations, enhancing data indexing at the storage layer and introducing new spatio-temporal processing methods, making it more applicable than traditional spatial Big Data systems.

*Spark-Based Big Data Systems* are gaining popularity due to their faster in-memory data processing capabilities compared to Hadoop-based solutions. GeoSpark [44] is a cluster computing framework that enables large-scale spatial data processing. It comprises a Spark Layer, a spatial RDD (SRDD) Layer, and a Spatial Query Processing Layer. These layers allow for storing geometrical and spatial objects (such as Point, LineString, Polygons, and Rectangles) and performing basic geometrical operations. Furthermore, spatial indexes (e.g., R-tree, Quad-tree) can be created to improve the performance of spatial data processing in each SRDD partition. STARK [45] presents a comprehensive solution for supporting spatio-temporal operations within the Spark framework. It provides spatial partitioners, different indexing modes, filter, join, and clustering operators for spatio-temporal data. Furthermore, unlike GeoSpark's SRDDs, which can only hold geometries of one type, STARK's RDDs can accommodate all kinds of columns, including spatial and temporal data. This allows for efficient processing of data in subsequent steps. Additionally, STARK extends Piglet's capabilities with an extended Pig Latin dialect for further ease of use. It also integrates seamlessly with the Spark API, eliminating users needing to adopt a separate API. STARK still needs to support more spatio-temporal queries than GeoSpark.

*Flink-Based Big Data Systems* are well-suited to dealing with real-time STBD due to their scalability and ability to process streaming data. GeoFlink [46] is a framework that integrates Flink with streaming spatial data (e.g., points, line strings, polygons) using a grid-based index. It currently supports spatial range, spatial $k$NN, and join queries but is limited to point data. GeoFlink comprises two core layers: the Spatial Stream Layer and the Real-time Spatial Query Processing Layer. The former converts incoming data streams into spatially referenced streams, while the latter enables spatial queries to be executed on these streams in real time. **SPEAR** [47] is a system for integrating temporal data into Flink. SPEAR supports streaming temporal point, line, and polygon data queries. It extends Flink by providing partitioning based on GeoHash, spatial and temporal indexing, and a library of spatio-temporal streaming processes. SPEAR enables dynamic query management in a distributed environment over high-velocity STBD streams, allowing fast real-time query responses.

## C. Data Analysis Layer

The data analysis layer comprises diverse spatio-temporal platforms tailored for AI training, recognizing the need to depart from traditional serial analytical algorithms that fall short of meeting the real-time processing demands of STBD. Integrating AI and GIS has enabled remarkable improvements in GIS's image processing and analysis capabilities and predictive power. This has facilitated the application of AI-GIS technology in various fields, such as ecological assessment, environmental protection, and agriculture. In recent years, the rapid advancement of AI and GIS has resulted in the emergence of several AI-GIS platforms. This section will provide a more in-depth examination of the features of each platform.

*1) Basic AI System:* TensorFlow, Keras, PyTorch, and MXNet [48] are the four most pertinent comparative platforms. In Table VI, the comparison results are presented.

*TensorFlow [49]*, a robust tool for tasks like image recognition, semantic segmentation, and natural language processing,

TABLE VI
COMPARISON OF AI FRAMEWORKS

| Category | TensorFlow | Keras | PyTorch | MXNet |
|---|---|---|---|---|
| Core Language | C++, Python | Python | C++, Python | C++, Python, R, Scala |
| Synchronous Mode | All-Reduce | All-Reduce | All-Reduce | Parameter Server |
| Asynchronous Mode | Parameter Server | Parameter Server | None | Parameter Server |
| Data structure | Static | Static | Dynamic | Static, Dynamic |
| Communication | gRPC | gRPC | Gloo | custom RPC |
| Multi GPU | Y | Y | Y | Y |
| Programming Paradigm | Imperative | Imperative | Imperative | Imperative, Declarative |
| Speed | Fast | Slow | Common | Very fast |

consists of two core modules: a static computational graph generator and a software library. Moreover, TensorFlow adeptly manages CPU, GPU, and memory resources. The solution, which involves the coordination of CPU and GPU management, enhances the efficiency of the training stage (or long-term processing) by implementing static scheduling and load-balancing algorithms instead of dynamic provisioning [50].

*Keras [51]* is not an independent AI platform but a lightweight application interface for TensorFlow. It highly integrates the functions of TensorFlow, has a low learning cost, and effectively calls existing functions of TensorFlow. However, it has a relatively slow running speed and strict code modification requirements and is only suitable for more straightforward application scenarios.

*PyTorch [52]* is a popular DL framework for handling dynamic computational graphs and performing various derivatives. It implements a custom allocator that incrementally builds up a CUDA memory cache and reassigns it to later allocations without using any CUDA APIs. By configuring the code, it is possible to use multiple GPUs for parallel processing, yet this can lead to a GPU memory imbalance problem. Specifically, the model and gradient lost during the calculation are saved on card 0 by default, causing card 0 to consume more video memory than the other cards [53].

*MXNet [54]* offers symbolic and imperative programming, enabling maximum efficiency and productivity. Its upper interface, Gluon, stands out from other AI platforms because it supports flexible, dynamic, and efficient static graphs. Additionally, MXNet has been designed to focus on distributed computing, providing superior support for multi-machine training. Moreover, MXNet utilizes object substitution, memory sharing, dynamic memory allocation, and other technologies, resulting in lower video memory consumption for most model training than others [55].

*2) Spatio-Temporal AI-GIS:* The traditional Big Data system presents certain limitations regarding GIS data processing. **Communication resources** can be a bottleneck, as GPUs may have enough processing power to handle image data in GIS. However, bandwidth and data transfer limitations impede further improvements in processing speed. Furthermore, the complexity of a model requires more parameters, and synchronizing them in a distributed system requires a considerable amount of bandwidth [56]. Additionally, **computing resources** can be an issue, as the system will allocate resources such as excessive CPU and GPU to a running task, resulting in lower resource utilization. The current resource scheduling algorithms need to be optimized explicitly for AI and, thus, cannot fully exploit the advantages of DL and Big Data systems when processing STBD [57].

For communication resources, two standard communication models are commonly employed: (1) a master-slave-based architecture [58], wherein a parameter server is maintained to store all parameters and devices communicate exclusively with the parameter server; and (2) a peer-to-peer (P2P)-based architecture [59], wherein devices interact with each other, resulting in an increased bandwidth overhead if the number of devices increases. To demonstrate the effectiveness of ML techniques for accelerating the processing of GIS data models in large datasets, the stale synchronous parallel (SSP) mechanism of TensorFlow can be utilized. This approach is designed to parallelize computations for large-scale data analysis, allowing for more efficient and faster processing. The traditional bulk synchronous parallel (BSP) mechanism assumes that all tasks are trained and updated with uniform parameters, leading to longer running times due to the different input parameters of each task. However, the SSP mechanism of TensorFlow allows faster-running tasks to commence the next iteration earlier with the parameters of the old version, thus improving the overall processing performance of the system and optimizing the GIS data model in a shorter duration.

STBD inherently involves data points characterized by spatial and temporal dimensions, exemplified by applications such as tracking the dynamic movement of objects over time [60]. Confronted with the formidable challenges posed by STBD, it becomes imperative to leverage new hardware architectures and employ effective parallel processing models to ensure efficient and real-time data analysis. Enhancing resource usage efficiency and reducing computational resource consumption is essential for achieving remarkable performance. To this end, an appropriate resource allocation approach should be employed to distribute resources to the relevant activities. Traditional neural networks are static models with numerous layers, each performing a designated task. TensorFlow proposes a concept based on dynamic flow control, which enables users to distribute data across multiple devices in a cluster. Each device independently calculates a part of the data, thus realizing the data parallelism function. Tiresias [61] is a GPU scheduler that works to minimize the execution time of individual tasks and prevent starvation tasks. It also considers the task's execution time and the GPU occupancy rate, calculates the task's service level, and prioritizes the tasks accordingly.

AI is crucial in GIS, particularly when managing unstructured data such as pictures and videos. DL offers an efficient solution for tasks such as target detection, binary classification, and feature classification. For instance, **DeepLabv3+** [62] is a TensorFlow-based tool that can perform semantic image segmentation, enabling us to differentiate between land, ocean, and buildings in satellite images. **Entropy-Weighted Network** [63] is another example used for polar sea ice open lead detection from synthetic-aperture radar images. In the past decade, road extraction from satellite images has been a popular research topic, with applications ranging from automated crisis response and road map updating to city planning, geographic information updating, and car navigation. **D-LinkNet** [64], a PyTorch-based high-resolution satellite imagery road extraction model, has been developed. In addition, **Orfeo ToolBox [65]** provides a processing library with AI algorithms that can be used to manage high-resolution optical, multispectral, and radar images at the TB scale. **PSGraph [66]**, a graph analytics system, has been developed using Spark executor and PyTorch for calculations and a distributed parameter server for storing frequently accessed models.

The above description highlights the essential role that AI technology can play in processing STBD on a Big Data system. For instance, the combination of TensorFlow and distributed systems can maximize the high performance of TensorFlow image processing and leverage the data parallelism of the distributed system. Furthermore, this block-based processing of Big Data can generate analytical models that can enhance the capabilities of problem-solving, automatic reasoning, decision-making, knowledge representation, and utilization, thereby enabling intelligent solutions to complex real-world problems. Depending on the specific AI methods applied to GIS applications, there are various combinations of GIS with expert or knowledge-based expert systems, pattern recognition, and decision support systems.

## *D. Cloud Native Layer*

The cloud native layer introduces the mainstream architecture of cloud scheduling systems, highlighting the need for comprehensive one-stop STBD platforms. The current challenge lies in the inefficiency of managing and scheduling data-intensive STBD and AI on existing cloud-native platforms. There is a pressing demand for a high-performance and scalable scheduling system tailored for cloud-native environments. The industry is actively working towards deploying Big Data platforms and AI training on cloud-native infrastructures to achieve excellent elasticity and scalability [67]. Despite these aspirations, there have been limited breakthroughs in this domain. The advent of container technologies and orchestration systems has significantly accelerated the development of cloud-native generation platforms. Notably, Docker and Kubernetes (K8s) technologies have emerged as the de facto standard for shaping the future of cloud-native architecture.

*1) New Cloud Computing Scheduling System:* STBD transfer involves intricate data transmission from the cloud to edge and terminal devices. Optimization techniques such as data segmentation and compression ensure efficient distribution and storage. Subsequently, leveraging edge computing effectively mitigates the transmission pressure on the cloud, enhancing overall data processing efficiency. Establishing a direct and efficient data transmission channel meets real-time processing requirements for a subset of STBD on terminal devices. Network optimization is achieved through the meticulous selection of transmission paths and the adoption of appropriate network protocols. These combined factors synergistically propel the efficient, secure, and real-time transmission of STBD within the Cloud-Edge-End environment.

A large-scale cloud platform often demands the efficient and reliable operation of tens of thousands of containers, necessitating a robust service orchestration system. Several cluster resource scheduling systems have emerged to meet the increasing resource demands from diverse services and tasks, including Borg, Mesos, and Omega [68]. Borg, a representative of centralized schedulers, has been a trailblazer with over a decade of production experience at Google, overseeing tasks such as GFS, BigTable, Gmail, and MapReduce. Mesos, representing two-level schedulers, facilitates the equitable sharing of cluster resources among multiple frameworks. As a shared-state scheduler example, Omega supports using different schedulers for various task types and handles resource request conflicts adeptly. K8s, an open-source project from Google designed for managing Docker clusters, inherits Borg's strengths and aims to orchestrate, deploy, run, and manage containerized applications. It features a unified scheduling framework capable of managing thousands of servers and tens of thousands of containers. It also includes a plugin interface for third-party customization and extension of new scheduling systems.

The industry adopts three primary strategies to address the challenges of developing and deploying STBD and AI applications in the cloud [69]. The first approach involves delivering cloud services through application- or service-level multitenancy. In this scenario, a database or AI service instance serves user needs, and user isolation is managed by the database or AI platform rather than the cloud infrastructure. Examples include AWS Aurora and Google Cloud AutoML. While flexible, this approach tends to be more application-specific and less versatile for general-purpose Big Data or database platforms. The second approach utilizes dedicated bare-metal machines within the cloud platform to provide Big Data or AI services, ensuring platform performance. However, this sacrifices elasticity and flexibility, making these bare-metal machines inefficient for scheduling and operational capabilities. The third approach confines the container platform to scheduling stateless applications, such as microservices, while deploying Big Data and AI platforms on traditional virtualization-based clouds. Although this guarantees robust scheduling capabilities, it compromises the stability of Big Data or AI platforms. While straightforward, real-world instances have demonstrated limited success due to stability concerns.

*2) One-Stop Spatio-Temporal Platform:* The advancement of GIS technology has enabled the development of one-stop GIS platforms that leverage Big Data systems and AI frameworks on cloud platforms for rapid analysis and processing. Such platforms can be used more efficiently in geoscience and social activities. This section compares the most representative one-stop spatio-temporal platforms in Table VII.

The Esri platform is the launching point for ArcGIS [70], a powerful GIS software that can be integrated into web servers and desktop applications. ArcGIS's programmable components span a wide range of objects, from fine-grained individual geometries to coarse-grained objects. With object pools, users can specify a maximum number of services to be deployed, allowing for more excellent compatibility with Big Data platforms such as Spark. ArcGIS Server also provides granular control over

TABLE VII
COMPARISON OF ONE-STOPGIS PLATFORMS

| Platform | ArcGIS | SuperMap | MapInfo | QGIS |
|---|---|---|---|---|
| Database type | PostgreSQL, PostGIS | PostgreSQL | PostgreSQL, PostGIS MySQL | SpatialLite, SQLite MySQL |
| Encryption | N | Y | N | N |
| Multiple dataset | N | Y | N | N |
| Data compression | N | Y | N | Y |
| Open-source | N | N | N | Y |
| Big data | Y | Y | N | plugin |
| AI | Y | Y | N | plugin |
| Cloud-native | Partial | Y | N | N |

the deployment of GIS services, allowing users to launch and cancel individual services. In contrast, SuperMap services must be activated or stopped, making granular management impossible. For cloud-native applications, cloud-based computing and storage are viable alternatives, although external cloud service providers such as AWS and Azure may be necessary.

SuperMap [71] is well-known for its capability to encrypt data using the SDB format, allowing storage of multiple datasets in a single source, encompassing simple points, lines, faces, or complex geometries. Its data type capabilities surpass ArcGIS's, as it adeptly handles vector files from CAD applications and raster data from picture formats such as PNG and IMG. SuperMap seamlessly integrates with Big Data tools and frameworks, including distributed file systems, spatio-temporal databases, and spatial analysis and processing tools like Spark. Its self-hosted cloud platform based on K8s enables GIS functions to be modularized into elastically scalable microservices, providing edge GIS tools for establishing a centralized cloud platform.

MapInfo [72] contains a powerful relational DBMS that organizes data in two-dimensional tables, incorporating graphical columns to contain graphical objects and standard data types for easy manipulation of graphical data. It also provides ODBC compatibility for various relational databases, ensuring the continuity of the original database and access to remote databases. Moreover, MapInfo supports the standard picture graphics format and a variety of others, including AutoCAD-generated DWG and DXF files.

QGIS [73] is a lightweight, open-source software package that offers a wide range of powerful mapping-related functions and is highly scalable. It includes shared GIS libraries such as GDAL and SQLite, which support more data formats than those supported by ArcGIS. Furthermore, QGIS can integrate with other desktop GIS software, such as GRASS GIS and SAGA GIS, allowing it to perform typical data processing and geographical analysis more effectively. However, it has some drawbacks; for instance, it is less stable than ArcGIS and can only merge two layers simultaneously. Compared to ArcGIS and SuperMap, QGIS is highly customizable and supports Big Data systems and AI algorithms. Nevertheless, it could benefit from being more feature-rich and more stable.

One-stop shops facilitate large-scale analysis of STBDs, yet they need to improve data security, scalability, and cloud-native support. Moreover, these platforms only offer cloud platform processing without focusing on optimizing efficiency through effective resource management. So, further research and development are needed.

## IV. RESOURCE MANAGEMENT

The seamless orchestration of computational power, storage, and networking resources expedites STBD processing and guarantees that platforms unlock their full potential, providing timely insights and actionable results in the dynamic realm of spatio-temporal data analytics.

### A. Resource Management in DBMS-GIS

Traditional DBMSs focus on efficiently creating, deleting, modifying, and querying data. With the addition of STBD and new storage devices, we can examine the computing resource management in DBMS-GIS systems from the perspectives of application workloads and hardware resources.

*1) Workload-Oriented Resource Management:* Diverse workloads exhibit unique resource requirements and sensitivity levels. We can skillfully manage resources to optimize performance by comprehensively understanding the application's features.

*Online transaction processing (OLTP)* and *onLine analytical processing (OLAP)* are distinctly different in terms of data processing technology. OLTP requires fast I/O performance due to soild consistency, supported by conflict resolution at write time (using read and write locks), leading to scalability issues [74]. On the other hand, OLAP focuses on data analysis and is akin to a data warehouse, so DBMSs do not need to be concerned with data consistency constantly; instead, they should focus on decision support and providing clear and intelligible query results, which are CPU- and memory-sensitive workloads [75]. Additionally, real-time data is another essential workload type. For instance, Ma et al. [76] utilized taxi passengers' real-time ride requests to schedule the proper taxis to pick them up. Streaming data will continually arrive at the DBMS in a fine-grained form, necessitating the need to balance indexing, processing, storage, I/O, and CPU consumption.

With the increasing popularity of NoSQL databases, more and more application scenarios are being uncovered. For example, some applications prioritize read performance, while others prioritize low-conflict writing, making resource management more complex.

*2) Storage-Oriented Resource Management:* The distinguishing attributes of STBD include its vast scale, notable diversity, and temporal dynamics. Consequently, storage systems must exhibit exceptional scalability and adaptability to effectively manage the continuous influx of data streams. STBD often integrates data from various sources and modalities, such as sensors and social media. Consequently, storage management systems require robust support for diverse data types and flexible models. Moreover, meeting the analytical and querying demands of STDB involves operations spanning diverse temporal and spatial scales. Therefore, storage systems must employ efficient spatio-temporal indexing and querying techniques for rapid data access and analysis. The storage engine comprises two essential components, disk, and memory, which determine the categorization of databases as disk-based or in-memory. Both components can significantly enhance system performance by reducing the amount of data that needs to be transferred between the CPU and traditional storage.

*Disk storage* is a service unit that utilizes a class of storage media (e.g., SSD) to facilitate computation. All data within the database is stored on disk, while memory is mainly used as a buffer for bulk writes. Sequential access is more efficient than random access due to the physical structure of the hard disk, so disk-based databases typically employ indexes to sort records in multiple fields, thus reducing data query time. To create these indexes, various data structures, such as B-trees, hash tables, B+ trees, and more, can be utilized [77]. Although disk storage may affect system performance, many databases still choose it as their primary storage due to its ability to ensure data consistency and integrity through on-disk log files and transaction management in relational databases, as outlined by the principles of atomicity, consistency, isolation, and durability [78]. Furthermore, disk storage is preferred due to its low cost, significantly lower than other storage devices in the storage hierarchy. There are two techniques to increase the throughput of disk-based systems to combat the subpar I/O performance of hard disk drives. (1) The number of hard disks and data processing parallelism can be increased to reduce the strain on each drive. New RAID architectures such as Elastic-RAID [79] can increase system throughput by several or tens, and data can be quickly rebuilt during a disk failure. Biscuit [80] is a near-storage computing framework for modern solid-state drives, allowing for distributed execution of data-intensive applications. (2) Individual disk performance can be improved. The 3-D NAND Flash Value-Aware SSD [81] is a reliable solid-state disk with high-speed access and low power consumption, which can enhance the performance of DBMS. Kang et al. [82] proposed the Smart SSD model, which uses an object-based communication protocol to exploit the processing power of SSDs.

*In-memory storage* integrates computational resources into innovative storage units with high-bandwidth connectivity, enhancing data read and write performance, and distributed architectures can significantly augment their capacity. Despite their imperfections, in-memory databases constitute an essential subset of distributed databases that are increasingly pivotal for modern data processing. The rapid evolution of distributed in-memory storage has paved the way for the emergence of in-memory data management systems for task processing and data analysis (e.g., HBase, Pig, and H-Store), catering to the demands of massive data processing. However, this technology has notable drawbacks, including cost, limited capacity, and potential data instability. The new non-volatile memory (NVM) storage has the characteristics of I/O performance close to DRAM, allowing faster and more flexible data transmission between memory, disk storage, and the CPU. Byte addressing fills the gap between traditional disk storage and memory, making NVM an essential tool to overcome the memory wall. Recent developments in NVM devices, such as resistive memory [83], spin-transfer torque random access memory, and Intel Optane DC persistent memory [84] have further narrowed this gap. Current research in NVM storage technology focuses on the issues of reading and writing speed differences, writing lifespan, and fault tolerance when accessing NVM storage devices from the perspectives of hardware structure design, reading and writing strategies, wear balance strategies, and fault tolerance strategies. For instance, NJS [85], an NVM-based file system with a write-efficient journaling scheme, reduces the logging overhead of traditional file systems and takes advantage of the byte-accessibility characteristic of NVM. Additionally, circ-Tree [86], a B+ tree variant with a circular design for NVM, enhances the access performance of Key-Value stores.

The implementation of functions reveals the difference between the two storage engines. In-memory databases are relatively straightforward, as writing and releasing random memory space is simple. However, disk-based databases must contend with more complex procedures, such as data referencing, file serialization, and defragmentation, which can be challenging to implement.

*3) Resource Management on Cloud: Storage optimization:* Cloud computing demands flexible storage capabilities as a service paradigm to ensure efficient data access. Q [87] introduced a dynamic shared memory management framework. This approach permits multiple virtual machines to dynamically access shared memory resources based on their requirements, enhancing inter-VM communication efficiency and VM memory swapping efficiency. Centaur [88] achieved dynamic partition caching for each VM through local replacement in each partition. This strategy results in lower cache miss rates and improved performance isolation and control for VM workloads, ensuring the efficiency and effectiveness of storage.

*Load Balancing:* To enhance the concurrent processing capacity of cloud environments, ensuring stable throughput and response times necessitates comprehensive management of machine workloads. H et al. [89] proposed a CMLB cloud-related multimedia system. It presented a highly effective load-balancing technique. It considers the load of all servers and the network conditions, effectively addressing resource scheduling and allocation issues. Zhao et al. [90] approached the challenge from a different angle, focused on optimizing the selection of physical hosts to improve the direction of task requests. They propose a load-balancing method based on the LB-BC technique, which combines Bayesian algorithms with clusters to identify the most suitable physical hosts for executing workloads. This approach effectively maintains long-term load balance, reduces task failures, and significantly boosts throughput.

## B. Resource Management in BigData-GIS System

Mainstream Big Data dedicated clusters and cloud computing platforms often require assistance to address complex diversification and hardware heterogeneity issues.

*1) Task Scheduling for Mixed Workloads: Task-DAG Schedulers* are devised to allocate resources to interdependent tasks within a job, considering the overall task-DAG structure. However, achieving an optimal schedule that minimizes the makespan while considering task dependencies and diverse resource demands is challenging. Spear [91] is a scheduling framework geared towards reducing the manufacturing time of complex jobs while considering both task dependencies and heterogeneous resource requirements. Decima [92], conversely, is a system that utilizes reinforcement learning and neural networks to generate workload-specific scheduling algorithms, eliminating the need for detailed instructions beyond a high-level objective. Lastly, DelayStage [93] introduces a stage delay scheduling strategy that interleaves cluster resources in parallel stages, thereby improving cluster resource utilization and enhancing job performance.

*2) Cluster-Oriented Multi-Resource Management: Harvest CPU resource:* Modern data analytics typically run tasks on statically reserved resources, prone to over-provisioning to guarantee quality of service (QoS), resulting in many resource time fragments. As such, the resource utilization of a data analytics cluster is severely underutilized. Shenango [94] offered comparable latencies while achieving far greater CPU efficiency. It reallocates cores across applications at a fine granularity of 5Âμs, enabling cycles that would otherwise have been wasted by latency-sensitive applications to be used productively by batch processing applications. Para [95] is an event-driven scheduling mechanism designed to harvest the CPU time fragments in co-located Big Data analytic workloads. Additionally, it identifies the idle CPU time window associated with each CPU core by capturing the task-switch event.

*Cluster Resource Scheduler:* Although cluster schedulers have significantly improved resource allocation, the utilization of the allocated resources could be higher due to inaccurate resource requests. Elasecutor [96] is a novel executor scheduler for data analysis systems that dynamically allocates and explicitly adjusts resources to executors over time based on predicted temporal/varying resource requirements. Ursa [97] enabled the scheduler to capture accurate resource demands dynamically from the execution runtime and to provide timely, fine-grained resource allocation based on monotasking. In addition, Flinkcl [98] extends the capabilities of Flink from CPU clusters to heterogeneous CPU-GPU clusters, significantly increasing its computational power.

*3) Resource Management on Cloud: Cloud resource utilization:* Resource underutilization is a pervasive issue in cloud data centers, which can be addressed by deploying multiple workloads in one cluster. To improve resource utilization while avoiding contention and performance interference, TPShare [99] proposed a simple yet effective vertical label mechanism to coordinate the time-sharing or space-sharing schedulers in different layers. Furthermore, the Scavenger [100] system was designed to function without offline profiling or prior information about the tenants' workloads to ensure the QoS of latency-sensitive services and reduce the performance impact on batch jobs. Additionally, the CERES [101] system was developed to guarantee the QoS of latency-sensitive services and minimize the performance impact on batch jobs.

*Containerized Resource Management:* ML workloads such as DL and hyperparameter tuning are computationally intensive and require parallel execution to reduce the learning time. KubeRay [102], an operator and suite of tools designed and built to create Ray clusters in K8s, provides an effective solution to meet this need with minimal effort.

## C. Resource Management in AI-GIS

This section focuses on two core aspects: accelerating AI training and improving cluster resource utilization for DL workload types and heterogeneous cluster resources.

*1) Application-Oriented Workload Types: Deep Neural Networks (DNNs)* continuously grow in size to improve the accuracy and quality of the models. Nevertheless, these strategies often result in suboptimal parallelization performance. Several research works have been proposed to accelerate distributed computing of DNNs. DNNs have seen remarkable growth in size to improve models' accuracy and quality. However, this has led to suboptimal parallelization performance. To address this issue, researchers have proposed various initiatives to expedite the distributed computation of DNNs [103]. These initiatives range from algorithmic modifications to hardware and infrastructure optimization. Algorithmic techniques such as model parallelism, data parallelism, and federated learning have been proposed to enable distributed computation of DNNs. Moreover, special-purpose processors, distributed memory architectures, and high-speed interconnects have been proposed as hardware solutions to enhance the efficiency of distributed computation [104]. Additionally, cloud computing, edge computing, and container orchestration have been proposed as infrastructure solutions to provide an optimal platform for distributed computation [105]. All of these initiatives have the potential to accelerate the distributed computation of DNNs significantly.

As the burgeoning trend of graph-based DL, **graph neural network (GNN)** have demonstrated exceptional capability in generating high-quality node feature vectors. However, existing one-size-fits-all GNN implementations must keep up with the ever-evolving GNN architectures, ever-increasing graph sizes, and diverse node embedding dimensionality. To address this challenge, GNNAdvisor [106] is an adaptive and efficient runtime system designed to expedite various GNN workloads on GPU platforms. Moreover, DistGNN [107] optimizes the Deep Graph Library for full-batch training on CPU clusters, featuring an efficient shared memory implementation, communication reduction with a minimum vertex-cut graph partitioning algorithm, and communication avoidance with a family of delayed-update algorithms.

The **transformer algorithm** has revolutionized the field of NLP in recent years. Unlike Recurrent Neural Network models, transformers can process sequences of any length in parallel, improving accuracy when dealing with long sequences. However, deploying efficient online services in data centers equipped with GPUs is still a challenge. Fang et al. [108] developed TurboTransformers, a transformer serving system composed of a computing runtime and a serving framework. ET [109] proposed a novel self-attention architecture and an attention-aware pruning design.

*2) Cluster-Oriented Resource Management: GPU Sharing Strategy:* GPUs currently stand as the predominant accelerators for DL applications. However, a significant drawback arises from the fact that many existing cluster schedulers treat GPUs as non-shareable devices. This results in certain tasks needing to fully occupy the resources of a single GPU, consequently diminishing the resource utilization rate of the entire cluster and influencing the average completion time of tasks [110]. An effective GPU-sharing strategy should maximize resource sharing while ensuring isolation between tasks. To achieve optimal resource sharing, both GPU computing resources and GPU storage resources should be shared to the greatest extent possible. This approach allows for accommodating more tasks on a single GPU. Simultaneously, to guarantee isolation, efforts should be made to minimize the impact between tasks, thereby reducing the influence on execution time and overall throughput [111].

*Heterogeneous Clusters Scheduling:* The advancement of hardware technology has made heterogeneous devices in clusters inevitable. Consequently, allocating all hardware devices as the same resource will lead to an inevitable loss of efficiency. To address this issue, it is necessary to explore resource scheduling strategies based on heterogeneous clusters, analyze the execution efficiency of different tasks on different hardware devices, and optimally allocate resources to avoid unnecessary waste [112].

*3) Resource Management on Cloud: Container Technology:* Nowadays, the most widely used cluster management solution is K8s, based on Docker containers. However, the technology for running AI training tasks in the Docker environment still needs to be developed. Moreover, these cluster management solutions cannot sense the underlying hardware, reducing distributed training efficiency and underutilizing cluster GPUs [113], [114]. To optimize the execution time of DL in Docker containers and improve the utilization of GPU clusters, Yeh et al. [115] proposed KubeShare, an extension of K8s that enables GPU sharing with fine-grained allocation. In addition, kube-Knots [116] can dynamically harvest spare compute cycles through dynamic container orchestration, allowing for the co-location of latency-critical and batch workloads while improving overall resource utilization. In FaST-GShare [117], an advanced spatio-temporal GPU sharing framework crafted for serverless computing in DL inferences, the FaST-Manager takes center stage. This strategic component adeptly limits and isolates spatio-temporal resources, optimizing GPU multiplexing efficiency.

### D. Resource Management in Cloud-GIS System

This section discusses the research progress of traditional cloud computing technologies. Subsequently, the focus is on recent research hotspots in cloud-native systems.

*1) Traditional Cloud Computing Technologies:* The current landscape of cloud services extends beyond distributed computing, encompassing distributed computing, utility computing, load balancing, parallel computing, network storage, hot backup redundancy, and virtualization. While traditional cloud computing has matured over the years, the emergence of new tools and frameworks has prompted further research and optimization.

Efficient resource management and task allocation under a certain hardware resource are critical for achieving significant performance improvements in the cloud. Li et al. [118] introduced the Load Balancing Ant Colony Optimizations algorithm that reduces task set generation time while maintaining scheduling flexibility. Kumar et al. [119] enhanced scheduling performance by combining min–min, and max–min methods with a standard genetic algorithm, improving traditional scheduling performance and resource utilization.

*2) Cloud Computing 2.0 Based on Cloud-Native:* cloud-native is wholly managed by a third party, is event-driven, stateless, and briefly kept in a compute container, whereas traditional cloud computing requires manual infrastructure building. Serverless deployment applications, derived from the benefits of cloud-native architectures, can automatically construct, deploy, and initiate services without additional infrastructure building.

*Serveless:* To enhance response time and decrease latency while optimizing costs through serverless resource scheduling, current strategies are centered on minimizing cold start time or reducing the frequency of cold starts. Cloud functions typically operate on pre-allocated virtual machines and containers, allowing for swift launches. However, functions reliant on extensive packages exhibit slower starts, diminishing the application's resilience and responsiveness to sudden increases in load. Abad et al. [120] introduced a package-aware scheduling algorithm to mitigate this challenge by assigning functions with similar package requirements to the same worker nodes. This approach amplifies packet caching hit rates, consequently reducing the latency of cloud functions. Additionally, the load on worker nodes is monitored to prevent it from surpassing a configurable threshold. Building on this, Oakes et al. [121] proposed a shared package cache to expedite the start-up time of cloud functions further, resulting in a substantial reduction in response time and latency.

*Virtualization:* Due to the unique characteristics of serverless applications, such as bursty, variable, and stateless behavior, existing platform scheduling mechanisms often need to catch up. The prevailing strategy is to enhance resource utilization dynamically and optimally, given these challenges. In pursuit of this goal, Kaffes et al. [122] introduced a centralized approach to eliminate queue imbalance, mitigate interference with core-based granularity, and propose a centralized kernel granularity scheduler. This design offers the advantage of directly assigning ready-to-execute functions to idle working cores, minimizing unnecessary queuing, eliminating queue imbalances and improving overall platform resource utilization. In a parallel effort, Jiang et al. [123] presented a hybrid job assignment approach to enhance workflow resource utilization. Their approach considers the resource consumption patterns of various workflow execution phases, culminating in the proposal of a serverless workflow execution system. The outcome is significantly more efficient execution of large-scale scientific workflows on public clouds than the traditional cluster execution model in extensive large-scale tests.

*Containerization:* The scale of containerization technology often involves hundreds or even thousands of servers, leading to a noticeable performance impact due to bandwidth and communication overhead between networks. There can be network bandwidth bottlenecks in scenarios with bursty workloads, particularly when hundreds of virtual machines hosting cloud functions utilize the same container images. To address this

TABLE VIII
REPRESENTATIVE SPATIO-TEMPORAL BIG DATA DATASET

| Classification | Dataset | Source | Application field |
|---|---|---|---|
| Basic ST data | Collection1 | NASA | Disaster Prediction, Mineral Exploration |
| | ASDC | NASA | Environmental protection, Weather forecast |
| | RSC11 | CAS | Weather forecast |
| | SIRI-WHU | WHU | Environmental protection, Image recognition |
| | RSSCN7 | WHU | Environmental protection, Weather forecas |
| | WHU-RS19 | WHU | Image recognition, Data analysis |
| Public thematic data | LandScan | ORNL | Risk assessment, Rescue planning |
| | Forex Dataset | Forex | Financial investment |
| | Sberbank | Kaggle | Economic regulation |
| IoT real-time perception data | KITTI | KIT | Intelligent driving |
| | Udacity Dataset | Udacity | Intelligent driving |
| | Mapillary Vistas | Mapillary | Image Identification |
| Internet online acquisition data | AmazonReviews | Amazon | Recommended system |
| | FilmTrust | FilmTrust | Recommended system |
| | Yelp | Yelp | Natural language processing |

challenge, Wang et al. [124] introduced FaaSNet, a lightweight, adaptable function tree structure that facilitates scalable container provisioning. Thomas et al. [125] concentrated on the total startup time, encompassing the duration required for initiating and establishing connections among a group of containers. Their proposed particle model, albeit compromising isolation, significantly reduces the interconnection time of containers.

## V. APPLICATIONS

Geoscience, a broad phrase that encompasses specific subjects such as climatology, hydrology, and others, is separate from the sphere of social activity, spanning a spectrum from epidemiology to commercial businesses. Geosciences are mostly concerned with offline examination of RS data and structured information gathered from sensors. Social activity revolves around the real-time study of geographic data generated by human activities.

It divides into two segments in the setting of STBD: one reflecting the physical environment and the other repeating the rhythms of human-social activity. To demonstrate the breadth and diversity of this data ecosystem, we've compiled a list of 15 open-source datasets relevant to STBD, which is shown succinctly in Table VIII.

### A. Geographic Environment

As Earth observation technology advances, there has been a remarkable increase in the volume of RS data. *The basic STBD* comprises a myriad of types: from vector and image data to geographic entity details, place names, address data, 3D model datasets, and mapping product information, all paired with their relevant metadata.

The multifaceted realm of geoscience calls for a robust, user-centric cloud platform to address the eclectic requirements of its practitioners. Acknowledging this, Varouchakis et al. [126] explored groundwater resource distribution, leveraging structured numerical datasets and classic modeling approaches. Concurrently, Le et al. [127] applied ConvLSTM to interpolate and prognosticate citywide air pollutants. While these endeavors capitalized on traditional analysis and processing modalities to frame databases and data processing infrastructure, the days of hands-on management could be better.

Today's environment, characterized by the synergy of Big Data and AI, heralds a transformative phase where integrated, one-stop platforms for data processing emerge as the norm. Ensuring data security and rigorous encryption, especially for delicate geographic datasets, stands central in this transformative journey. Addressing these imperatives will augment the efficacy and reach of geoscientific inquiry and catalyze groundbreaking applications and insights in the Earth observation domain.

### B. Human-Social Activities

The rapid progression of Internet technology and the ubiquity of social media platforms have led to a daily influx of vast spatio-temporal data rooted in human social interactions. Within this realm, *public thematic data* encompasses a spectrum ranging from population statistics and macroeconomic insights to point-of-interest data, geographical censuses, monitoring records, and their corresponding metadata. Parallelly, the realm of *IoT real-time perception data*–time-stamped data harnessed through advanced IoT sensing–embraces real-time spatio-temporal information from Earth observation sensor networks, specialized sensor readings, real-time industry-shared themes, and their associated metadata. Tapping into the vast web ocean, *internet online acquisition data* is primarily sourced using tools like web crawlers, capturing diverse online data streams.

Social activity data analysis is now at the forefront of research, with wearable sensors and Internet crawlers instrumental in accumulating STBD rich with user-generated content. Such data analyses offer profound insights for both social and commercial sectors. Castro et al. [128] adeptly used daily COVID-19 data to delineate spatio-temporal spread patterns across regions. For timely epidemiological assessments during public health crises, a seamless STBD flow is paramount. This calls for merging Big Data stream processing with AI predictive algorithms. Exploiting data distribution traits, expansive GIS analyses are undertaken. To heighten efficiency, computing resources are judiciously dispersed based on data's temporal and spatial nature. A sleek computational framework emerges by refining cache processes and curtailing node communication overhead.

In intelligent transportation, STBD has showcased pivotal advancements. Real-time storage and indexing of vehicular patterns and traffic data ensure swift spatio-temporal query responses for traffic oversight. Immediate, extensive video and sensor data analysis aids in swift traffic event detection and resolution. An accurate traffic prediction model emerges with DL algorithms, refining traffic light coordination and boosting traffic flow efficiency [129]. Cloud-native platforms have proven invaluable in intelligent transportation implementations across cities [130], fostering cross-city data collaboration, offering versatile services to traffic authorities, and realizing optimal resource exploitation.

## VI. Research Agenda

In prior sections, we delved into STBD and its processing platforms, encompassing foundational attributes, spatio-temporal nuances, and resource management intricacies. This segment elucidates STBD's potential in amplifying asset management efficiency across varied platforms, bolstering processing speed. Concurrently, we introduce emerging research avenues, providing in-depth insights and recommendations for future exploration.

### *A. Research Agenda for DBMS-GIS*

This section examines how a compute-storage separation design affects remote data access. We improve dynamic memory consumption by combining local caching with cloud storage, forming a training data management system to increase efficiency during learning. We investigate the links between data storage strategies and training costs in the emerging Cloud 2.0 ecosystem, characterized by various metering schemes. We present a cost-cutting data storage technique for multi-cloud scenarios, resulting in significant consumer savings.

*1) Dynamic Demand Based on Cross-Task Data Sharing System:* DL training in large high-performance computing clusters involves numerous nodes working together to read and process tasks. While data shuffling between training phases is common in DL jobs, direct data synchronization among multiple computer features is the best technique. Serverless models, such as AWS Lambda, present issues in effective inter-function communication and consistent data state retention between calls because of their limited memory and storage. To address this, we propose using shared local storage to collect intermediate data during cloud-based DL training, eliminating the requirement for external data persistence mechanisms like as databases, cloud disks, or Redis clusters and avoiding duplicate data transfers. We want to create a high-performance intermediary storage system that emphasizes dynamic data-sharing strategies and allows for smooth data synchronization across several job functions. We explore DL workload communication properties deeper to carefully hone low-latency, high-throughput inter-task data exchanges. Our ultimate goal is to create a scalable hybrid cloud storage infrastructure that combines cloud-centric data management with local storage, allowing real-time data sync via concurrent cloud storage operations.

*2) High-Performance Local Cache Method Based on Data Reuse Frequency Prediction:* DL's voracious appetite for data necessitates notable storage throughput. This is centralized in container-supported computing functions, where thousands of serverless functions actively engage in intensive shared file system operations. This situation leads to increased latency, erratic read/write performance, and challenges in managing the rapid scaling of cloud-native apps, particularly during peak traffic periods. We're designing a robust local cache management system to address this, accounting for data access patterns, latency, and volume.

Our blueprint highlights two cache scenarios for training data: A dynamic cache file selection strategy for optimizing storage in skewed read/write ratios scenarios is unveiled. This involves real-time file access pattern analysis, determining if a requested file merits cache placement. This avoids caching files accessed only once and adjusts for frequently accessed files based on temporal patterns. A generational tech-based cache file update strategy is proposed in instances with an extensive cache footprint during training. This flexible approach enables tailoring with diverse cache replacement algorithms for optimal access performance. Concludingly, acknowledging that data significance varies across learning stages–typically high initially and tapering over time–we integrate lifecycle management for cached objects. This adjusts cache object lifecycles and predicts cache durations and access patterns, refining cache replacement while upholding its intrinsic strategies.

*3) Cost-Aware High-Performance Hybrid Storage Solution in Multi-Cloud Environments:* Stateless computational functions, while agile, grapple with challenges in preserving data locally, a pivotal facet for DL training. While the cache from Task 2 augments throughput and trims latency, it compounds both caching and training expenses. Moreover, its absence of automated storage oversight mandates cloud storage integration for expansive data training and enduring data preservation. Notably, cloud storage providers vary in service quality and pricing. The versatility of multi-cloud storage–leveraging multiple providers based on cost-effectiveness and niche services–enhances user satisfaction, cuts costs, and sidesteps vendor dependency. Without a judicious storage strategy, there's a tangible risk of squandering cloud resources in this landscape. Therefore, an astute storage approach is paramount for cost efficiency and resource conservation in cloud environments.

### *B. Research Agenda for BigData-GIS*

This section examines the fluctuating computational and data requirements during DL training phases. We introduce an adaptive power management strategy tailored to enhance the efficacy of modern cloud-based training systems. Considering the diverse pricing structures in the Cloud 2.0 epoch, we delve into the relationship between computing power distribution and overall training expenses. This leads us to formulate a cost-optimized data storage algorithm for multi-cloud contexts to reduce user costs.

*1) Dynamic Computing Resource Allocation Based on Real-Time Resource Requirements:* DL exhibits pronounced variations in resource consumption, highlighting the need for swift resource provisioning at training onset and efficient resource release post-training. Given that resource demands, data processing, and duration vary across training and inference tasks, there's a pressing need for platforms with elastic resource provisioning. As the number of training iterations rises, most models demand fewer resources. Hence, there's a need for resource allocation that dynamically adjusts to optimize usage as training progresses. Unlike traditional systems where users are constrained by static configurations, our proposed system introduces a dynamic resource model for learning training. This model ensures full resource utilization by adjusting allocations per iteration and rebalancing resources based on diverse strategies and task priorities. This leads to an enhanced scheduling mechanism and augmented system throughput.

*2) Task Scheduling Based on Data Dependency and Network Topology Awareness:* In breaking down DL training tasks into serverless functions, each function intertwines with a series of dependent services. Changes in runtime parameters or serverless function scheduling can ripple through this dependency chain. We want to add the scheduler's data dependency awareness to address this. This enhancement will be achieved by interfacing with a global parameter configuration center, allowing the system to dynamically gauge each event's impact on every training subtask. The system can judiciously choose the optimal scheduling strategy by evaluating potential state transition graphs, minimizing resource wastage.

Moreover, on many function-as-a-service platforms, functions are encapsulated within containers, which run atop either virtual or physical hardware. While a server node may house multiple container services, the container's view is restricted to its network, devoid of the underlying machine's IP awareness. This must be clarified to understand data distribution's physical layout, affecting computational and data locality. We'd suggest embedding a network physical topology module in the scheduling system to reduce this. This module would dynamically map physical IPs to container IPs. Hence, when stateful services are scheduled, they can reference this mapping to identify an available physical machine. This ensures that computational tasks are routed based on the physical data storage layout, leveraging data locality for performance enhancement.

*3) Cost-Estimated Elastic Configuration for Computational Functions:* The era of Cloud 2.0 has seen serverless computing revolutionize resource management, bringing enhanced pricing models, granular resource metrics, and significant benefits, especially for DL training architectures. Given the periodic fluctuations of DL workloads and short-term unpredictable load spikes, shaping an adaptive resource configuration for emerging cloud functionalities is a complex task.

We propose a cost-centric computational resource allocation strategy tailored for intensive DL contexts to navigate these challenges. Central to this strategy is the judicious selection of billing methods. In essence, serverless computing platforms typically offer two payment modes: Prepaid Resources At a lower unit price. Postpaid Resources are priced higher per unit. Aligning the right payment mode with the application's resource consumption pattern can yield substantial cost benefits. Our approach hinges on real-time serverless computing usage monitoring, underpinned by stable and elastic computational resource usage prediction models. Using this foundation, the strategy dynamically tweaks the prepaid resource ratio, ensuring optimal cost benefits by favoring the cheaper option wherever possible.

A responsive approach is favored for short-term, unpredictable, and bursty computational demands. Using feedback-controlled adaptive resource management, our method swiftly reconfigures virtual resources across nodes, offering quick reactions to load alterations while adhering to service objectives. In contrast, our strategy leans on statistical and ML techniques for long-term, cyclical DL workloads that exhibit predictable patterns. These tools dive deep into system logs and extensive load change datasets to construct performance blueprints aligned with prolonged load trends. This offers a backbone for making informed decisions on cost-effective, large-scale computational configurations. Complementing this, we have crafted an adaptive minimum-cost framework for optimal function placement.

### C. Research Agenda for AI-GIS

This section discusses the new issues of scheduling DL training activities within GPU clusters. We provide a multi-tiered, granular cluster scheduler to bridge the gap between basic hardware and the complicated demands of upper-tier computing jobs. Anchored by GPU sharing frameworks and different resource clusters, our focus is divided into two main goals: accelerating DL training and maximizing cluster resource efficiency. The scheduler's design promotes GPU sharing paradigms, fine-tunes distributed training task distribution, and leverages the capabilities of various devices to accelerate training efforts.

*1) GPU Sharing Strategy and Fine-Grained Scheduling Resources:* Heterogeneous clusters amalgamate diverse hardware devices, offering computational prowess. These clusters proffer benefits like cost-efficient high-performance computing, robust scalability, and optimal computational resource use. They also cater to diverse computational architectures, ensuring holistic satisfaction. Yet, managing resources and choreographing tasks emerge as pressing issues.

To navigate these intricacies, we propose a nuanced scheduling paradigm tailored for clusters interspersing CPUs and an array of GPUs. This tripartite framework encapsulates a task scheduling model, a device evaluative mechanism, and a central scheduler. As an initial step, DL training tasks undergo preprocessing to glean vital metrics. Once extracted, these tasks segue into a system task reservoir, biding their time for scheduling. Simultaneously, the device evaluation mechanism periodically assesses the capabilities of each device. The scheduler, acting on real-time resource availability and task prerequisites, cherry-picks the most fitting task for execution from this reservoir.

*2) Scheduling Optimization Based on Heterogeneous Clusters:* GPUs, the cornerstone for accelerating DL training, are increasingly harnessed for real-time tasks. Many tasks may only tap into a fraction of a GPU's vast capabilities. Thus, the move towards multi-tasking on a singular GPU has gained momentum. Nvidia's Multi-Process Service (MPS) champions this cause by enabling concurrent task execution on a single GPU. However, its Achilles' heel is inter-task interference. Should one task falter (including premature termination), it cascades a ripple effect, leading to the failure of other cohabiting tasks on that GPU. This underscores the imperative for a failsafe GPU-sharing framework.

Our proposition pivots on a dual-faceted GPU-sharing model, integrating both spatial and time-sharing elements: **Spatial**

**Sharing:** In a scenario where myriad tasks operate concurrently on one GPU, MPS offers a rudimentary spatial sharing foundation, albeit with its limitations concerning task isolation. To address this, we delve deeper into the intricate architecture of the GPU. Tasks are then tethered to designated Streaming Multiprocessors and memory portions. This strategic allocation naturally engenders task segregation, sidestepping the pitfalls of MPS. **Time-Sharing:** This dimension focuses on temporally segmenting GPU execution and allotting tasks predicated on strategic metrics. To lay the groundwork, we first dissect the characteristics intrinsic to DL training tasks, including their GPU computational demand, memory footprint, and intercommunication intervals. A tailored scheduling framework emerges for our segmented GPU using these metrics as guiding beacons. Both methodologies effectively fragment the GPU into finer units, equipping subsequent scheduling algorithms with a granular toolset. The cluster's multitasking potential surges by doing so, allowing for concurrent task execution. This symbiosis not only amplifies task equity but also curtails task queuing durations and overall execution timelines.

*3) Optimization Strategy for Distributed Based on Container Technology.:* Distributed DL offers shortened training periods by dispersing computing needs over numerous nodes. However, the distribution's fundamental essence, model parameter synchronization, provides an inherent overhead. Communication between nodes, essential for ensuring model coherence, extends the time we want to save. As a result, the spatial positioning of these computational nodes emerges as a critical factor of distributed DL effectiveness.

Our strategy unfolds in strategic steps: (1) Task Characterization: The locational sensitivity of each DL task introduced is originally assessed. Due to the wide range of sensitivities, tasks are divided into separate groups. These identified metrics are saved in a special database. (2) Hardware Topology Understanding: Our database includes a comprehensive mapping of the hardware topology. This information reservoir serves as the North Star for later scheduling methods. Given its container-centric architecture, the current paradigm often exhibits shortsightedness, emphasizing the need for a deeper understanding of the underlying hardware. The inefficiency in training and inefficient GPU utilization that results is palpable. (3) Holistic Scheduling method: Our scheduling method is a wise blend of depth and dynamism, proposing a synthesis of task qualities and hardware topology. A multi-tiered analysis provides depth by addressing job peculiarities and the diverse hardware landscape. Dynamism derives from the realization that when resources are regained, fragmentation becomes a potential concern. Due to this, our plan incorporates regular recalibration, enabling us to continuously optimize task assignments. This frequent reshuffling reduces resource fragmentation while simultaneously improving the fluidity and efficacy of resource scheduling. This scheduling strategy, anchored in data-driven insights and characterized by adaptability, promises a harmonious marriage of computational and communicative demands and sets the stage for efficient and responsive distributed DL architectures.

## D. Research Agenda for Cloud-GIS

This section delves into the complexities associated with cloud-native computing in the context of next-generation Big Data processing and AI training. We introduce data management, resource distribution, and task coordination strategies. We aim to craft a robust theoretical framework, grounded in cloud-native service principles, that bolsters efficiency in cloud-centric Big Data processing and AI training. This framework promises marked improvements in computational throughput and cost-effectiveness. The implications of this approach are particularly pronounced in sectors like public safety, intelligent transport, and advanced healthcare.

*1) High-Performance Data Management Based on Stateless Services:* We propose a storage service built on local storage, tailored to efficiently back stateful services, including database services (e.g., MySQL) and distributed storage, within a stateless cloud-native framework. All storage resources are streamlined into a cloud storage pool in this setup. Each node's local storage is segmented into multiple persistent volume groups (PVGs). Containers can specifically request storage resources, termed as persistent volumes (PVs). Each PV, adaptable in size, can only be engaged by a single container. It supports both HDDs and SSDs and a tiered storage amalgamation. Stateful container services specify PV requirements (like size, IOPS, SSD, or HDD preferences) when initiating. The onus is on the scheduler to rapidly pinpoint an apt machine on the platform to accommodate the container. If any container falters, the scheduler identifies this malfunction, and leveraging the data's physical topology, earmarks a suitable machine for reboot.

STBD and AI training place considerable demands on storage throughput. As computational functions within container clusters centrally process vast amounts of data and trigger thousands of serverless services, they exert significant pressure on shared file systems with intensive read/write operations. Such a burden results in increased latency, spikes in high latency, compromised read/write stability, and challenges in managing peak traffic within short durations. To combat this, we propose a robust local cache management method factoring in data usage frequency, access expiration, and data size. We've conceptualized designs for two primary training data cache contexts: We propose a dynamic cache file selection strategy for scenarios experiencing low cache storage utilization due to minimal read/write ratios. This strategy undertakes real-time analysis of commonly accessed files' size and nature. It discerns if requested files warrant caching, eschewing single-use files from cache placement. Further, it factors in file access frequency and access time intervals, minimizing errors in cache decisions. We suggest a cache file update strategy rooted in generational technology to elevate efficiency in cache replacements during data training scenarios with extensive cached files. This method tailors different cache replacement algorithms to suit various high-performance access needs. Acknowledging that data's importance varies across learning stages–with heightened access initially that tapers over time–we envision amplifying our cache replacement strategy. We aim to integrate lifecycle management for cache objects.

We can update this lifecycle to predict data cache duration and access frequency, refining cache replacement execution while preserving its innate attributes.

*2) Dynamic Computing Resource Allocation Based on Real-Time Demand:* Cloud platform resources are logically pooled and segmented based on CPU, GPU, memory, storage, and network attributes. When scheduling a container, it specifies resource needs via orchestration files or configurations. Given the resource pool's status and host resources, the scheduler must pinpoint an apt physical node for the container in milliseconds. The scheduler's efficacy, speed, and balance influence cloud resource utilization in expansive cloud clusters. Existing schedulers in Big Data or AI platforms often need to improve with the unique demands of container applications, like their bursty nature, variability, and statelessness. Additionally, most current cloud-native schedulers focus primarily on the container platform, overlooking the diverse resource management needs of varying application loads. This oversight is especially glaring with the dynamic demands of STBD or AI training.

Our proposed solution emphasizes the real-time perception of cluster states. By harnessing data and computation topologies, we advocate for a cloud-native service model-based scheduling approach. The goal is dual: achieving cloud computing elasticity and optimizing deep learning computing performance. This entails real-time, precise resource allocation and control, promoting adaptive service scheduling. It moves away from static strategies, offering flexible computational power control and dynamic resource allocation during iterative computations.

STBD and AI training tasks have a pronounced resource consumption pattern, exhibiting notable peaks and troughs. It's essential to quickly allocate computing resources during the initial phase of data training and efficiently reclaim them post-training to optimize resource use. Given the diverse resource demands of data processing, model training, and inference, there's a pressing need for computing platforms with adaptable resource provisioning. As the number of training iterations for Big Data or AI models rises, their resource needs typically wane. Thus, resources for ML training must be adjustable in real time to optimize utilization throughout the task. Unlike traditional systems, where users set fixed configurations upfront, our proposed system introduces a dynamic resource consumption model for training tasks. It caters to each iteration's specific needs, ensuring full resource utilization. Furthermore, dynamically rebalancing resource allocation based on strategies and job priorities augments the scheduler's efficiency and amplifies overall system throughput.

*3) Scheduling Computation Tasks Based on Data Topology:* Containers in cloud-native platforms designed for flexible and elastic scheduling often function within dedicated network spaces that remain concealed from the host network. Because a single server node might hold numerous container services, this configuration creates a disparity: more container IPs than host IPs. Big data or AI frameworks such as MapReduce, Spark, and TensorFlow have historically determined a service's launch location based on the physical topology of data storage. However, in a containerized environment, jobs see only the container network, necessitating additional computation and data localization.

To address this, we've built a network physical topology module inside the scheduler. This module keeps a live map of physical IPs and container IPs. As a result, when scheduling a stateful service, it can use this map to find the best physical location. Hosts in a containerized cloud environment can function as standalone networks, each hosting one or more sub-networks. These sub-networks provide the network addresses for containers. When a container is closed, its address is returned to the pool and made available for future allocations.

Our application-aware data topology approach unfolds as follows: (1) When a containerized data application (like a Spark application) initiates a computational task, the scheduler identifies the container required (such as an HDFS data node) using dependency relationships. (2) The scheduler queries the physical node's network and storage details where the container resides via the metadata module. This module, updated in real time, fetches the latest metadata from storage and network services. (3) Given that distributed storage often holds multiple replicas, the scheduler uses anti-affinity rules to ensure distribution across different physical nodes. This results in a list of potential physical nodes. (4) After considering each node's resource consumption and load, the scheduler opts for the least loaded node to deploy the associated computing service container. (5) Since the computing and storage services coexist on the same host, they can employ local networking or domain socket mechanisms to expedite data transfer and boost performance.

## VII. Conclusion

This article provides a comprehensive analysis of STBD processing platforms emphasizing resource management. We explore the evolution and key applications of STBD, segmenting its analytical ecosystem into DBMS-GIS, BigData-GIS, AI-GIS, and Cloud-GIS. While our primary focus is resource management, we also introduce emerging research areas. Our in-depth review aims to offer fresh insights for researchers, fostering the enhancement and optimization of STBD systems. Overall, our findings pave the way for scholars striving to deepen their grasp on the STBD analytical landscape.

## References

[1] Z. Jiang and S. Shekhar, "Spatial and spatiotemporal big data science," *Spatial Big Data Sci.: Classification Techn. Earth Observ. Imagery*, pp. 15–44, 2017.

[2] H. Ahmed and M. A. Ismail, "A structured approach towards big data identification," *IEEE Trans. Big Data*, vol. 9, no. 1, pp. 147–159, Feb. 2023.

[3] F. Li, J. Zhang, W. Guo, R. Xian, Y. Wu, and Y. Song, "Research and practice on the whole process management and sharing technology system of multi-source heterogeneous spatio-temporal data," in *Proc. IEEE 29th Int. Conf. Geoinformatics*, 2022, pp. 1–6.

[4] Y. Dildar Korkmaz, "Evaluating the convergence of high-performance computing with big data, artificial intelligence and cloud computing technologies," Master's thesis, Dept. Inf. Syst., Middle East Technical University, 2023.

[5] D. Soni and N. Kumar, "Machine learning techniques in emerging cloud computing integrated paradigms: A survey and taxonomy," *J. Netw. Comput. Appl.*, vol. 205, 2022, Art. no. 103419.

[6] X. Li, H. Liu, W. Wang, Y. Zheng, H. Lv, and Z. Lv, "Big data analysis of the Internet of Things in the digital twins of smart city based on deep learning," *Future Gener. Comput. Syst.*, vol. 128, pp. 167–177, 2022.

[7] P. Galetsi, K. Katsaliaki, and S. Kumar, "Big data analytics in health sector: Theoretical framework, techniques and prospects," *Int. J. Inf. Manage.*, vol. 50, pp. 206–216, 2020.

[8] N. Deepa et al., "A survey on blockchain for big data: Approaches, opportunities, and future directions," *Future Gener. Comput. Syst.*, vol. 131, pp. 209–226, 2022.

[9] R. Mortaheb and P. Jankowski, "Smart city re-imagined: City planning and GeoAI in the age of big data," *J. Urban Manage.*, vol. 12, no. 1, pp. 4–15, 2023.

[10] F. Baig, "High performance spatial and spatio-temporal big data processing," Ph.D. dissertation, Dept. Comput. Sci., State University of New York at Stony Brook, 2021.

[11] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends," *Swarm Evol. Computation*, vol. 62, 2021, Art. no. 100841.

[12] S. Duan et al., "Distributed artificial intelligence empowered by end-edge-cloud computing: A survey," *IEEE Commun. Surv. Tut.*, vol. 25, no. 1, pp. 591–624, First Quarter 2023.

[13] W. Song, L. Wang, Y. Xiang, and A. Y. Zomaya, "Geographic spatiotemporal big data correlation analysis via the Hilbert–Huang transformation," *J. Comput. Syst. Sci.*, vol. 89, pp. 130–141, 2017.

[14] A. Eldawy and M. F. Mokbel, "The era of big spatial data," in *Proc. 31st IEEE Int. Conf. Data Eng. Workshops*, 2015, pp. 42–49.

[15] C. Yang, Q. Huang, Z. Li, K. Liu, and F. Hu, "Big data and cloud computing: Innovation opportunities and challenges," *Int. J. Digit. Earth*, vol. 10, no. 1, pp. 13–53, 2017.

[16] W. Liu et al., "Cluster analysis of microscopic spatio-temporal patterns of tourists' movement behaviors in mountainous scenic areas using open GPS-trajectory data," *Tourism Manage.*, vol. 93, 2022, Art. no. 104614.

[17] D. Li, S. Wang, and D. Li, *Spatial Data Mining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.

[18] C. Liu, F. Wang, W. Wang, Q. Zhang, and L. Zhang, "Study on mountain space in mount Tai region based on GIS spatial data analysis," in *Proc. IEEE Conf. Telecommun., Opt. Comput. Sci.*, 2021, pp. 400–403.

[19] X. Wang et al., "Spatio-temporal analysis and prediction of cellular traffic in metropolis," *IEEE Trans. Mobile Comput.*, vol. 18, no. 9, pp. 2190–2202, Sep. 2019.

[20] M. Sakr, C. Ray, and C. Renso, "Big mobility data analytics: Recent advances and open problems," *GeoInformatica*, vol. 26, no. 4, pp. 541–549, 2022.

[21] D. Liu, P. Xu, and L. Ren, "TPFlow: Progressive partition and multidimensional pattern extraction for large-scale spatio-temporal data analysis," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 1, pp. 1–11, Jan. 2019.

[22] W. Khan, T. Kumar, C. Zhang, K. Raj, A. M. Roy, and B. Luo, "SQL and NoSQL database software architecture performance analysis and assessments—a systematic literature review," *Big Data Cogn. Comput.*, vol. 7, no. 2, 2023, Art. no. 97.

[23] G. Malewicz et al., "Pregel: A system for large-scale graph processing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2010, pp. 135–146.

[24] Y. Long, E. Lee, D. Kim, and S. Mukhopadhyay, "Q-PIM: A genetic algorithm based flexible DNN quantization method and application to processing-in-memory platform," in *Proc. 57th ACM/IEEE Des. Automat. Conf.*, 2020, pp. 1–6.

[25] E. Vermij, L. Fiorin, C. Hagleitner, and K. Bertels, "Sorting big data on heterogeneous near-data processing systems," in *Proc. Comput. Front. Conf.*, 2017, pp. 349–354.

[26] D. Ribeiro de Almeida, C. de Souza Baptista, F. Gomes de Andrade, and A. Soares, "A survey on big data for trajectory analytics," *ISPRS Int. J. Geo-Inf.*, vol. 9, no. 2, 2020, Art. no. 88.

[27] D. Guo and E. Onstein, "State-of-the-art geospatial information processing in NoSQL databases," *ISPRS Int. J. Geo-Inf.*, vol. 9, no. 5, 2020, Art. no. 331.

[28] R. Li et al., "Just: JD urban spatio-temporal data engine," in *Proc. IEEE 36th Int. Conf. Data Eng.*, 2020, pp. 1558–1569.

[29] R. Li et al., "TrajMesa: A distributed NoSQL-based trajectory data management system," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 1013–1027, Jan. 2023.

[30] S. Nishimura, S. Das, D. Agrawal, and A. El Abbadi, "MD-HBase: A scalable multi-dimensional data infrastructure for location aware services," in *Proc. IEEE 12th Int. Conf. Mobile Data Manage.*, 2011, pp. 7–16.

[31] J. N. Hughes, A. Annex, C. N. Eichelberger, A. Fox, A. Hulbert, and M. Ronquest, "GeoMesa: A distributed architecture for spatio-temporal fusion," in *Geospatial Informatics, Fusion, and Motion Video Analytics V*, vol. 9473. Bellingham, WA, USA: SPIE, 2015, pp. 128–140.

[32] J. K. Nidzwetzki and R. H. Güting, "Distributed secondo: A highly available and scalable system for spatial data processing," in *Proc. Int. Symp. Spatial Temporal Databases*, 2015, pp. 491–496.

[33] J. K. Nidzwetzki and R. H. Guting, "BBoxDB-a scalable data store for multi-dimensional big data," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 1867–1870.

[34] J. Qin, L. Ma, and J. Niu, "THBase: A coprocessor-based scheme for big trajectory data management," *Future Internet*, vol. 11, no. 1, 2019, Art. no. 10.

[35] J. Wang, C. Xu, J. Zhang, and R. Zhong, "Big data analytics for intelligent manufacturing systems: A review," *J. Manuf. Syst.*, vol. 62, pp. 738–752, 2022.

[36] I. Polato, R. Ré, A. Goldman, and F. Kon, "A comprehensive view of hadoop research—a systematic literature review," *J. Netw. Comput. Appl.*, vol. 46, pp. 1–25, 2014.

[37] S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin, "A survey of open source tools for machine learning with big data in the Hadoop ecosystem," *J. Big Data*, vol. 2, no. 1, 2015, Art. no. 24.

[38] S. Tang, B. He, C. Yu, Y. Li, and K. Li, "A survey on spark ecosystem: Big data processing infrastructure, machine learning, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 71–91, Jan. 2022.

[39] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink: Stream and batch processing in a single engine," *Bull. IEEE Comput. Soc. Tech. Committee Data Eng.*, vol. 36, no. 4, pp. 28–38, 2015.

[40] F. Hueske and V. Kalavri, *Stream Processing With Apache Flink: Fundamentals, Implementation, and Operation of Streaming Applications*. Sebastopol, CA, USA: O'Reilly, 2019.

[41] M. M. Alam, L. Torgo, and A. Bifet, "A survey on spatio-temporal data analytics systems," *ACM Comput. Surv.*, vol. 54, no. 10s, pp. 1–38, 2022.

[42] A. Eldawy and M. F. Mokbel, "SpatialHadoop: A MapReduce framework for spatial data," in *Proc. IEEE 31st Int. Conf. Data Eng.*, 2015, pp. 1352–1363.

[43] L. Alarabi, "ST-Hadoop: A MapReduce framework for big spatio-temporal data," in *Proc. ACM Int. Conf. Manage. Data*, 2017, pp. 40–42.

[44] J. Yu, J. Wu, and M. Sarwat, "GeoSpark: A cluster computing framework for processing large-scale spatial data," in *Proc. 23rd SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2015, pp. 1–4.

[45] S. Hagedorn, P. Gotze, and K.-U. Sattler, "The stark framework for spatio-temporal data analytics on spark," *Datenbanksysteme für Bus., Technologie und Web*, vol. P-265, pp. 123–142, 2017.

[46] S. A. Shaikh, K. Mariam, H. Kitagawa, and K.-S. Kim, "GeoFlink: A distributed and scalable framework for the real-time processing of spatial streams," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 3149–3156.

[47] F. Baig, D. Teng, J. Kong, and F. Wang, "Spear: Dynamic spatio-temporal query processing over high velocity data streams," in *Proc. IEEE 37th Int. Conf. Data Eng.*, 2021, pp. 2279–2284.

[48] A. Jain, A. A. Awan, H. Subramoni, and D. K. Panda, "Scaling TensorFlow, PyTorch, and MXNet using MVAPICH2 for high-performance deep learning on frontera," in *Proc. IEEE/ACM 3rd Workshop Deep Learn. Supercomput.*, 2019, pp. 76–83.

[49] M. Abadi et al., "{TensorFlow }: A system for { Large-Scale} machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Des. Implementation*, 2016, pp. 265–283.

[50] L. Baresi, G. Quattrocchi, and N. Rasi, "Resource management for TensorFlow inference," in *Proc. Int. Conf. Service-Oriented Comput.*, 2021, pp. 238–253.

[51] A. Gulli and S. Pal, *Deep Learning With Keras*. Packt Publishing Ltd, 2017.

[52] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," *Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 8024–8035, 2019.

[53] M. Wahib et al., "Scaling distributed deep learning workloads beyond the memory capacity with karma," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2020, pp. 1–15.

[54] T. Chen et al., "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015, *arXiv:1512.01274*.

[55] Y. Gao et al., "Estimating GPU memory consumption of deep learning models," in *Proc. 28th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2020, pp. 1342–1352.

[56] Q. Zheng et al., "BiPS: Hotness-aware bi-tier parameter synchronization for recommendation models," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2021, pp. 609–618.

[57] A. Krisilias, N. Provatas, N. Koziris, and I. Konstantinou, "A performance evaluation of distributed deep learning frameworks on CPU clusters using image classification workloads," in *Proc. IEEE Int. Conf. Big Data*, 2021, pp. 3085–3094.

[58] Q. Yang et al., "A distributed swarm optimizer with adaptive communication for large-scale optimization," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3393–3408, Jul. 2020.

[59] A. Kumari, A. Shukla, R. Gupta, S. Tanwar, S. Tyagi, and N. Kumar, "ET-DeaL: A P2P smart contract-based secure energy trading scheme for smart grid systems," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2020, pp. 1051–1056.

[60] S. Ke, Z. Pan, T. He, Y. Liang, J. Zhang, and Y. Zheng, "AutoSTG+: An automatic framework to discover the optimal network for spatio-temporal graph prediction," *Artif. Intell.*, vol. 318, 2023, Art. no. 103899.

[61] J. Gu et al., "Tiresias: A GPU cluster manager for distributed deep learning," in *Proc. 16th USENIX Symp. Netw. Syst. Des. Implementation*, 2019, pp. 485–500.

[62] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 801–818.

[63] Z. Liang, X. Pang, Q. Ji, X. Zhao, G. Li, and Y. Chen, "An entropy-weighted network for polar sea ice open lead detection from sentinel-1 SAR images," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–14, 2022.

[64] L. Zhou, C. Zhang, and M. Wu, "D-LinkNet: LinkNet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 182–186.

[65] M. Grizonnet, J. Michel, V. Poughon, J. Inglada, M. Savinaud, and R. Cresson, "Orfeo toolbox: Open source processing of remote sensing images," *Open Geospatial Data, Softw. Standards*, vol. 2, no. 1, pp. 1–8, 2017.

[66] J. Jiang et al., "PSGraph: How tencent trains extremely large-scale graphs with spark?," in *Proc. IEEE 36th Int. Conf. Data Eng.*, 2020, pp. 1549–1557.

[67] S. Wang, Y. Zhong, and E. Wang, "An integrated GIS platform architecture for spatiotemporal big data," *Future Gener. Comput. Syst.*, vol. 94, pp. 160–172, 2019.

[68] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Gener. Comput. Syst.*, vol. 91, pp. 407–415, 2019.

[69] L. F. Bittencourt, A. Goldman, E. R. Madeira, N. L. da Fonseca, and R. Sakellariou, "Scheduling in distributed systems: A cloud computing perspective," *Comput. Sci. Rev.*, vol. 30, pp. 31–54, 2018.

[70] E. Pimpler, *Spatial Analytics With ArcGIS*. Packt Publishing Ltd, 2017.

[71] E. van Rees and R. Takken, "Supermap," *GeoInformatics*, vol. 14, no. 1, 2011, Art. no. 32.

[72] T. Probert, "Mapinfo professional," *GeoInformatics*, vol. 13, no. 6, 2010, Art. no. 62.

[73] A. Graser, *Learning QGIS 2.0*. Packt Publishing Ltd, 2013.

[74] N. Priya and E. Punithavathy, "A review on database and transaction models in different cloud application architectures," in *Proc. 2nd Int. Conf. Sustain. Expert Syst.*, 2022, pp. 809–822.

[75] Y. Zhang, X. Zhou, Y. Zhang, Y. Zhang, and S. Wang, "Virtual denormalization via array index reference for main memory OLAP," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 4, pp. 1061–1074, Apr. 2016.

[76] S. Ma, Y. Zheng, and O. Wolfson, "Real-time city-scale taxi ridesharing," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 7, pp. 1782–1795, Jul. 2015.

[77] R. Chopade and V. Pachghare, "MongoDB indexing for performance improvement," in *Proc. ICT Syst. Sustainability*, 2020, pp. 529–539.

[78] M. K. Iskander, T. Trainor, D. W. Wilkinson, A. J. Lee, and P. K. Chrysanthis, "Balancing performance, accuracy, and precision for secure cloud transactions," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 417–426, Feb. 2014.

[79] J. Yao, H. Jiang, Q. Cao, L. Tian, and C. Xie, "Elastic-RAID: A new architecture for improved availability of parity-based RAIDs by elastic mirroring," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 1044–1056, Apr. 2016.

[80] B. Gu et al., "Biscuit: A framework for near-data processing of big data workloads," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 153–165, 2016.

[81] Y. Deguchi, T. Nakamura, A. Hayakawa, and K. Takeuchi, "3-D NAND flash value-aware SSD: Error-tolerant SSD without ECCs for image recognition," *IEEE J. Solid-State Circuits*, vol. 54, no. 6, pp. 1800–1811, Jun. 2019.

[82] Y. Kang, Y.-S. Kee, E. L. Miller, and C. Park, "Enabling cost-effective data processing with smart SSD," in *Proc. IEEE 29th Symp. Mass Storage Syst. Technol.*, 2013, pp. 1–12.

[83] Y. Zhang, D. Feng, W. Tong, J. Liu, C. Wang, and J. Xu, "Tiered-ReRAM: A low latency and energy efficient tlc crossbar reram architecture," in *Proc. 35th Symp. Mass Storage Syst. Technol.*, 2019, pp. 92–102.

[84] P. Zardoshti, M. Spear, A. Vosoughi, and G. Swart, "Understanding and improving persistent transactions on optane$^{\text{TM}}$ DC memory," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2020, pp. 348–357.

[85] X. Zhang, D. Feng, Y. Hua, and J. Chen, "Optimizing file systems with a write-efficient journaling scheme on non-volatile memory," *IEEE Trans. Comput.*, vol. 68, no. 3, pp. 402–413, Mar. 2019.

[86] C. Wang, G. Brihadiswarn, X. Jiang, and S. Chattopadhyay, "Circ-tree: A B+-tree variant with circular design for persistent memory," *IEEE Trans. Comput.*, vol. 71, no. 2, pp. 296–308, Feb. 2022.

[87] Q. Zhang and L. Liu, "Shared memory optimization in virtualized cloud," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, 2015, pp. 261–268.

[88] R. Koller, A. J. Mashtizadeh, and R. Rangaswami, "Centaur: Host-side SSD caching for storage performance control," in *Proc. IEEE Int. Conf. Autonomic Comput.*, 2015, pp. 51–60.

[89] H. Wen, L. Chuang, Z. Hai-ying, and Y. Yang, "Effective load balancing for cloud-based multimedia system," in *Proc. IEEE Int. Conf. Electron. Mech. Eng. Inf. Technol.*, 2011, pp. 165–168.

[90] J. Zhao, K. Yang, X. Wei, Y. Ding, L. Hu, and G. Xu, "A heuristic clustering-based task deployment approach for load balancing using bayes theorem in cloud environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 305–316, Feb. 2016.

[91] Z. Hu, J. Tu, and B. Li, "Spear: Optimized dependency-aware task scheduling with deep reinforcement learning," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 2037–2046.

[92] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proc. ACM Special Int. Group Data Commun.*, 2019, pp. 270–288.

[93] W. Shao, F. Xu, L. Chen, H. Zheng, and F. Liu, "Stage delay scheduling: Speeding up dag-style data analytics jobs with resource interleaving," in *Proc. 48th Int. Conf. Parallel Process.*, 2019, pp. 1–11.

[94] A. Ousterhout, J. Fried, J. Behrens, A. Belay, and H. Balakrishnan, "Shenango: Achieving high CPU efficiency for latency-sensitive datacenter workloads," in *Proc. 16th USENIX Symp. Netw. Syst. Des. Implementation*, 2019, pp. 361–378.

[95] Y. Wang, H. Qu, J. Yu, and Z. Yu, "Para: Harvesting CPU time fragments in big data analytics," in *Proc. IEEE 14th Int. Conf. Cloud Comput.*, 2021, pp. 625–636.

[96] L. Liu and H. Xu, "Elasecutor: Elastic executor scheduling in data analytics systems," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 681–694, Apr. 2021.

[97] T. Jin, Z. Cai, B. Li, C. Zheng, G. Jiang, and J. Cheng, "Improving resource utilization by timely fine-grained scheduling," in *Proc. 15th Eur. Conf. Comput. Syst.*, 2020, pp. 1–16.

[98] C. Chen, K. Li, A. Ouyang, and K. Li, "FlinkCL: An OpenCL-based in-memory computing architecture on heterogeneous CPU-GPU clusters for big data," *IEEE Trans. Comput.*, vol. 67, no. 12, pp. 1765–1779, Dec. 2018.

[99] Y. Wang, L. Li, Y. Wu, J. Yu, Z. Yu, and X. Qian, "TPShare: A time-space sharing scheduling abstraction for shared cloud via vertical labels," in *Proc. ACM/IEEE 46th Annu. Int. Symp. Comput. Archit.*, 2019, pp. 499–512.

[100] S. A. Javadi, A. Suresh, and A. Gandhi, "Scavenger: A black-box batch workload resource manager for improving utilization in cloud environments," in *Proc. ACM Symp. Cloud Comput.*, 2019, pp. 272–285.

[101] J. Yu, D. Feng, W. Tong, and Y. Xiong, "CERES: Container-based elastic resource management system for mixed workloads," in *Proc. Int. Conf. Parallel Process.*, 2021, pp. 1–10.

[102] A. Kanso et al., "Designing a kubernetes operator for machine learning applications," in *Proc. 7th Int. Workshop Container Technol. Container Clouds*, 2021, pp. 7–12.

[103] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive DNN surgery for inference acceleration on the edge," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1423–1431.

[104] W. Rang, D. Yang, Z. Li, and D. Cheng, "Scalable data management on hybrid memory system for deep neural network applications," in *Proc. IEEE Int. Conf. Big Data*, 2021, pp. 1470–1480.

[105] C. Hu, H. H. Liang, X. M. Han, B. A. Liu, D. Z. Cheng, and D. Wang, "Spread: Decentralized model aggregation for scalable federated learning," in *Proc. 51st Int. Conf. Parallel Process.*, 2022, pp. 1–12.

[106] Y. Wang et al., “{GNNAdvisor }: An adaptive and efficient runtime system for { GNN} acceleration on {GPUs},” in *Proc. 15th USENIX Symp. Operating Syst. Des. Implementation*, 2021, pp. 515–531.
[107] V. Md et al., “DistGNN: Scalable distributed training for large-scale graph neural networks,” in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2021, pp. 1–14.
[108] J. Fang, Y. Yu, C. Zhao, and J. Zhou, “TurboTransformers: An efficient GPU serving system for transformer models,” in *Proc. 26th ACM SIG-PLAN Symp. Princ. Pract. Parallel Program.*, 2021, pp. 389–402.
[109] S. Chen et al., “ET: Re-thinking self-attention for transformer models on GPUs,” in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2021, pp. 1–18.
[110] S. Choi, S. Lee, Y. Kim, J. Park, Y. Kwon, and J. Huh, “Serving heterogeneous machine learning models on {Multi-GPU } servers with { Spatio-Temporal} sharing,” in *Proc. USENIX Annu. Tech. Conf.*, 2022, pp. 199–216.
[111] C. Hwang, T. Kim, J. Shin, and K. Park, “Elastic resource sharing for distributed deep learning,” in *Proc. 18th USENIX Symp. Netw. Syst. Des. Implementation*, 2021, pp. 721–739.
[112] G. Lim, J. Ahn, W. Xiao, Y. Kwon, and M. Jeon, “Zico: Efficient {GPU } memory sharing for concurrent { DNN} training,” in *Proc. USENIX Annu. Tech. Conf.*, 2021, pp. 161–175.
[113] D. Narayanan, K. Santhanam, F. Kazhamiaka, A. Phanishayee, and M. Zaharia, “Heterogeneity-Aware cluster scheduling policies for deep learning workloads,” in *Proc. 14th USENIX Symp. Operating Syst. Des. Implementation*, 2020, pp. 481–498.
[114] Y. Jiang, Y. Zhu, C. Lan, B. Yi, Y. Cui, and C. Guo, “A unified architecture for accelerating distributed {DNN} training in heterogeneous {GPU/CPU} clusters,” in *Proc. 14th USENIX Symp. Operating Syst. Des. Implementation*, 2020, pp. 463–479.
[115] T.-A. Yeh, H.-H. Chen, and J. Chou, “KubeShare: A framework to manage GPUs as first-class and shared resources in container cloud,” in *Proc. 29th Int. Symp. High-Perform. Parallel Distrib. Comput.*, 2020, pp. 173–184.
[116] P. Thinakaran, J. R. Gunasekaran, B. Sharma, M. T. Kandemir, and C. R. Das, “Kube-knots: Resource harvesting through dynamic container orchestration in GPU-based datacenters,” in *Proc. IEEE Int. Conf. Cluster Comput.*, 2019, pp. 1–13.
[117] J. Gu, Y. Zhu, P. Wang, M. Chadha, and M. Gerndt, “FaST-GShare: Enabling efficient spatio-temporal GPU sharing in serverless computing for deep learning inference,” in *Proc. 52nd Int. Conf. Parallel Process.*, 2023, pp. 635–644.
[118] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, “Cloud task scheduling based on load balancing ant colony optimization,” in *Proc. IEEE 6th Annu. ChinaGrid Conf.*, 2011, pp. 3–9.
[119] P. Kumar and A. Verma, “Scheduling using improved genetic algorithm in cloud computing for independent tasks,” in *Proc. Int. Conf. Adv. Comput. Commun. Inform.*, 2012, pp. 137–142.
[120] C. L. Abad, E. F. Boza, and E. Van Eyk, “Package-aware scheduling of FaaS functions,” in *Proc. ACM/SPEC Int. Conf. Perform. Eng.*, 2018, pp. 101–106.
[121] E. Oakes, L. Yang, K. Houck, T. Harter, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, “Pipsqueak: Lean lambdas with large libraries,” in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. Workshops*, 2017, pp. 395–400.
[122] K. Kaffes, N. J. Yadwadkar, and C. Kozyrakis, “Centralized core-granular scheduling for serverless functions,” in *Proc. ACM Symp. Cloud Comput.*, 2019, pp. 158–164.
[123] Q. Jiang, Y. C. Lee, and A. Y. Zomaya, “Serverless execution of scientific workflows,” in *Proc. Int. Conf. Service-Oriented Comput.*, 2017, pp. 706–721.
[124] A. Wang et al., “FaaSNet: Scalable and fast provisioning of custom serverless container runtimes at alibaba cloud function compute,” in *Proc. USENIX Annu. Tech. Conf.*, 2021, pp. 443–457.
[125] S. Thomas, L. Ao, G. M. Voelker, and G. Porter, “Particle: Ephemeral endpoints for serverless networking,” in *Proc. 11th ACM Symp. Cloud Comput.*, 2020, pp. 16–29.
[126] E. A. Varouchakis, P. G. Theodoridou, and G. P. Karatzas, “Spatiotemporal geostatistical modeling of groundwater levels under a Bayesian framework using means of physical background,” *J. Hydrol.*, vol. 575, pp. 487–498, 2019.
[127] V.-D. Le, T.-C. Bui, and S.-K. Cha, “Spatiotemporal deep learning model for citywide air pollution interpolation and prediction,” in *Proc. IEEE Int. Conf. Big Data Smart Comput.*, 2020, pp. 55–62.
[128] M. C. Castro et al., “Spatiotemporal pattern of covid-19 spread in brazil,” *Science*, vol. 372, no. 6544, pp. 821–826, 2021.
[129] X. Yin, G. Wu, J. Wei, Y. Shen, H. Qi, and B. Yin, “Deep learning on traffic prediction: Methods, analysis, and future directions,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4927–4943, Jun. 2022.
[130] M. Shengdong, X. Zhengxian, and T. Yixiang, “Intelligent traffic control system based on cloud computing and big data mining,” *IEEE Trans. Ind. Informat.*, vol. 15, no. 12, pp. 6583–6592, Dec. 2019.

**Huanghuang Liang** received the BS and MS degrees in automation engineering from the Anhui University of Technology in 2016 and the University of Electronic Science and Technology of China in 2019. He is currently working toward the PhD degree in computer science with Wuhan University. His research interests include cloud computing and Big Data systems.

**Zheng Zhang** is currently working toward the graduate degree with the School of Computer Science, Wuhan University. His research interests are distributed DL model training and deployment, DNN network optimization.

**Chuang Hu** (Member, IEEE) received the BS and MS degrees from Wuhan University in 2013 and 2016, and the PhD degree from the Hong Kong Polytechnic University in 2019. He is currently an associate researcher with the School of Computer Science at Wuhan University. His research interests include edge learning, federated learning/analytics, and distributed computing.

**Yili Gong** (Member, IEEE) received the BS degree in computer science from Wuhan University in 1998, and the PhD degree in computer architecture from the Institute of Computing, Chinese Academy of Sciences in 2006. She is currently an Associate Professor with the School of Computer Science at Wuhan University. Her interests include intelligent operations and maintenance in HPC environments, distributed file systems and blockchain systems.

**Dazhao Cheng** (Senior Member, IEEE) received the BS and MS degrees in electrical engineering from the Hefei University of Technology in 2006 and the University of Science and Technology of China in 2009, and the PhD degree from the University of Colorado at Colorado Springs in 2016. He was an AP with the University of North Carolina at Charlotte in 2016-2020. He is currently a professor with the School of Computer Science, Wuhan University. His research interests include Big Data and cloud computing.