VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

HO CHI MINH UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

SPECIALIZE PROJECT REPORT

BUILDING AN INTEGRATED DATABASE FOR SHORT TERM FORECASTING SYSTEMS IN HYDRO-METEOROLOGY

Major: Computer Science

THESIS COMMITTEE:	**Council 5**
SUPERVISORS:	**Lê Hồng Trang**
	Trương Quỳnh Chi
REVIEWER:	**Lê Thị Bảo Thu**
	—o0o—
STUDENT 1:	**Trần Hà Tuấn Kiệt (2011493)**
STUDENT 2:	**Nguyễn Đức Thuỵ (2012158)**

Ho Chi Minh City, 1/2024

Declaration

We guarantee that this research is our own, conducted under the supervision and guidance of Assoc. Prof. Dr. Lê Hồng Trang. The result of our research is legitimate and has not been published in any form prior to this. All materials used within this research are collected by ourselves, by various sources and are appropriately listed in the references section. In addition, within this research, we also used the results of several other authors and organizations. They have all been aptly referenced. In any case of plagiarism, we stand by our actions and are to be responsible for it. Ho Chi Minh City University of Technology therefore is not responsible for any copyright infringements conducted within our research

Ho Chi Minh City, Ngày 3 tháng 1 năm 2024

Team Members

Trần Hà Tuấn Kiệt

Nguyễn Đức Thuỵ

Acknowledgments

We, the research group consisting of two members, would like to send our thanks toward everyone who has contributed to the success of this thesis.

First and foremost, we would like to express our gratitude to Assoc. Prof. Dr. Lê Hồng Trang . The dedication and extensive knowledge of our supervisor not only guided us through the challenges of the project but also helped us develop research skills and critical thinking.

We also want to extend special thanks to all the friends, colleagues, and family who supported us throughout the research process. The input and opinions of everyone enriched the content and quality of the project.

Last, but certainly not least, we appreciate each other - research partners accompanying us in every step of the project. Our collaboration and joint contributions have resulted in a product that we take pride in.

We believe that this project marks a significant step in our development and could not have been achieved without the support and contributions of everyone involved.

Abstract

In this document, we introduce a comprehensive proposal to integrate, coordinate, and monitor meteorological and hydrological data in Vietnam. The proposed system can serve as a foundation for building a national database specializing in meteorological information.

To clarify our proposal, we have built a comprehensive pilot version, in which we have simulated the process of collecting and transmitting data from the weather station at Nha Be. At the same time, we focused on the process of converting and organizing data storage to ensure transparency and optimal performance in information processing.

This research is a manifestation of significant efforts in optimizing the management of meteorological data in Vietnam. The proposed system not only addresses the challenges of integration and coordination but also lays the foundation for a robust national database, serving the diverse needs of research and applications in the field of meteorology.

Bảng phân công công việc

STT	Họ và tên	MSSV	Phân chia công việc
1	Trần Hà Tuấn Kiệt	2011493	- Exploratory Data Analysis - System proposal and architecture - Research and Develop with new technologies - Implement system endpoints
2	Nguyễn Đức Thuy	2012158	- Exploratory Data Analysis - System proposal and architecture - Research and Develop with new technologies - Implement data processing flows

Mục lục

Danh sách hình vẽ

Danh sách bảng

Chương 1

Introduction

1.1 Phát biểu bài toán

The National Center for Hydrometeorological Forecasting (NCHMF), abbreviated as "Trung tâm Dự báo khí tượng thuỷ văn quốc gia" in Vietnamese, is an organizational unit under the General Department of Meteorology and Hydrology, Ministry of Natural Resources and Environment[7]. The National Hydro-Meteorological Forecasting Center has several crucial missions, including the establishment and presentation of standards and technical regulations for meteorological and hydrological forecasting, the operation of the national forecasting and warning system, monitoring and reporting on weather conditions and climate change, issuing and disseminating forecast bulletins and warnings, and participating in international meteorological agreements. Additionally, the center is responsible for conducting research, application, and technology transfer related to forecasting and warning, and implementing administrative reform and anti-corruption measures. These key missions contribute significantly to the center's role in ensuring public safety and providing essential meteorological and hydrological information.

There is a deliberate focus on those aspects of climate data management that are of interest to NMHSs wishing to make the transition to a modern climate database management system and, just as important, on what skills, systems and processes need to be in place to ensure that operations are sustained. In the context of the ever-growing complexity of climate change, the task of creating an integrated database for short-term forecasting systems in the fields of meteorology and hydrology poses a considerable challenge. The question at hand is how we can optimize the management of weather information from multiple sources and store it efficiently in a database. This optimization is crucial to ensure the provision of synchronized and high-quality information to support forecasting systems.

1.2 Động lực thực hiện

Information about the weather has been recorded in manuscript form for many centuries. The early records included notes on extreme and, sometimes, catastrophic events and also on phenomena such as the freezing and thawing dates of rivers, lakes and seas, which have taken on a higher profile with recent concerns about climate change. Specific journals for the collection and retention of climatological information have been used over the last two or three centuries (WMO 2005). The development of instrumentation to quantify meteorological phenomena and the dedication of observers to maintaining methodical, reliable and well-documented records paved the way for the organized management of climate data. Since the 1940s, standardized forms and procedures gradually became more prevalent and, once computer systems were being used by NMHSs, these forms greatly assisted the computerized data entry process and consequently the development of computer data archives. The latter part of the twentieth century saw the routine exchange of weather data in digital form and many meteorological and related data centers took the opportunity to directly capture and store these in their databases. Much was learned about automatic methods of collecting and processing meteorological data in the late 1950s, a period that included the International Geophysical Year and the establishment of the World Weather Watch. The WMO's development of international guidelines and standards for climate data management and data exchange assisted NMHSs in organizing their data management activities and, less directly, also furthered the development of regional and global databases. Today, the management of climate records requires a systematic approach that encompasses paper records, microfilm/microfiche records and digital records, where the latter include image files as well as the traditional alphanumeric representation.

Before electronic computers, mechanical devices played an important part in the development of data management. Calculations were made using comptometers, for example, with the results being recorded on paper. A major advance occurred with the introduction of the Hollerith system of punch cards, sorters and tabulators. These cards, with a series of punched holes recording the values of the meteorological variables, were passed through the sorting and tabulating machines enabling more efficient calculation of statistics. The 1960s and 1970s saw several NMHSs implementing electronic computers and gradually the information from many millions of punched cards was transferred to magnetic tape. These computers were replaced with increasingly powerful mainframe systems and data were made available online through developments in disk technology.

Aside from advances in database technologies, more efficient data capture was made possible through the mid-to-late 1990s with an increase in automatic weather stations (AWSs), electronic field books (i.e. on-station notebook computers used to enter, quality control and transmit observations), the Internet and other advances in technology. Not surprisingly, there are a number of trends already underway that suggest there are many further benefits for NMHSs in managing data and servicing their clients. The Internet is already delivering greatly improved data access capabilities and, providing security issues are managed, we can expect major opportunities for data managers in the next five to ten years. In addition, Open Source7 relational database systems may also remove the cost barriers to relational databases for many NMHSs over this period.

1.3 Mục tiêu dự án

It is essential that both the development of climate databases and the implementation of data management practices take into account the needs of the existing, and to the extent that it is predictable, future data users. While at first sight this may seem intuitive, it is not difficult to envisage situations where, for example, data structures have been developed that omit data important for a useful application or where a data centre commits too little of its resources to checking the quality of data for which users demand high quality.

In all new developments, data managers should either attempt to have at least one key data user as part of their project team or undertake some regular consultative process with a group of user stakeholders. Data providers or data users within the organization may also have consultative processes with end users of climate data (or information) and data managers should endeavour to keep abreast of both changes in needs and any issues that user communities have. Put simply, data management requires awareness of the needs of the end users.

At present, the key demand factors for data managers are coming from climate prediction, climate change, agriculture and other primary industries, health, disaster/emergency management, energy, natural resource management (including water), sustainability, urban planning and design, finance and insurance. Data managers must remain cognizant that the existence of the data management operation is contingent on the centre delivering social, economic and environmental benefit to the user communities it serves. It is important, therefore, for the data manager to encourage and, to the extent possible, collaborate in projects which demonstrate the value of its data resource. Even an awareness of studies that show, for example, the economic benefits from climate predictions or the social benefits from having climate data used in a health warning system, can be useful in reminding senior NMHS managers or convincing funding agencies that data are worth investing in. Increasingly, value is being delivered through integrating data with application models (e.g. crop simulation models, economic models) and so integration issues should be considered in the design of new data structures.

1.4 Phạm vi dự án

The project will focus on Ho Chi Minh City, a large urban area with a unique climate and significant impact on the daily lives of the community.

We will conduct research and collect data from multiple meteorological and hydrological monitoring stations in the city to ensure diversity and representation of local weather conditions. Once we have gathered sufficient data, we will proceed with preprocessing and standardizing the data to ensure accuracy and consistency of the dataset. Finally, we will build an integrated database to store and manage information from various sources, creating a unified and reliable data source. The integrated information source will be used to provide multidimensional and detailed data on current weather conditions and short-term forecasts. The goal is to help the community and relevant entities better prepare for unpredictable weather fluctuations.

Hoaving an integrated database is expected to improve the effectiveness of weather forecasting models, enabling individuals and businesses to intelligently and safely cope with challenging weather conditions.

Chương 2

Cơ sở lý thuyết

2.1 Hệ Cơ sở dữ liệu

Database systems perform vital functions for all sorts of organizations because of the growing importance of using and managing data efficiently. A database system consists of a software, a database management system (DBMS) and one or several databases. DBMS is a set of programs that enables users to store, manage and access data. In other words database is processed by DBMS, which runs in the main memory and is controlled by the respective operating system

A database is a logically coherent collection of data with some inherent meaning and represents some aspects of the real world. A random assortment of data cannot be referred to as a database. Databases draw a sharp distinction between data and information. Data are known facts that can be recorded and that have implicit meaning. Information is data that have been organized and prepared in a form that is suitable for decision-making. Shortly information is the analysis and synthesis of data. The most fundamental terms used in database approach are ìentityî, ìattributeî and ìrelationshipî. An entity is something that can be identified in the usersí work environment, something that the users want to track. It may be an object with a physical or conceptual existence. An attribute is a property of an entity. A particular entity will have a value for each of its attributes. The attribute values that describe each entity become a major part of data stored in the database

Database Management System is a general-purpose software system designed to manage large bodies of information facilitating the process of defining, constructing and manipulating databases for various applications. Specifying data types, structures and constraints for the data to be stored in the database is called defining a database. Constructing the database is the process of storing data itself on some storage medium that is controlled by the DBMS. Querying to retrieve specific data, updating the database to reflect changes and generating reports from the data are the main concepts of manipulating a database. The DBMS functions as an interface between the users and the database ensuring that the data is stored persistently over long periods of time, independent of the programs that access it [9]. DBMS can be divided into three subsystems; the design tools subsystem, the run time subsystem and the DBMS engine.

The design tools subsystem has a set of tools to facilitate the design and creation of the database and its applications. Tools for creating tables, forms, queries and reports are components of this system. DBMS products also provide programming languages and interfaces to programming languages. The run time subsystem processes the application components that are developed using the design tools. The last component of DBMS is the DBMS engine which receives requests from the other two components and translates those requests into commands to the operating system to read and write data on physical media [6].

Database approach has several advantages over traditional file processing in which each user has to create and define files needed for a specific application. In these systems' duplication of data is generally inevitable causing wasted storage space and redundant efforts to maintain common data up-to date. In database approach data is maintained in a single storage medium and accessed by various users. The self-describing nature of database systems provides information not only about database itself but also about the database structure such as the type and format of the data. A complete definition and description of database structure and constraints, called meta-data, is stored in the system catalog. Data abstraction is a consequence of this self-describing nature of database systems allowing programdata independence. DBMS access programs do not require changes when the structure of the data files are changed hence the description of data is not embedded in the access programs. This property is called program-data independence. Support of multiple views of data is another important feature of database systems, which enables different users to view different perspective of database dependent on their requirements. In a multi-user database environment users probably have access to the same data at the same time as well as they can access different portions of database for modification. Concurrency control is crucial for a DBMS so that the results of the updates are correct. The DBMS software is to ensure that concurrent transactions operate correctly when several users are trying to update the same data

Using a DBMS also eliminates unnecessary data redundancy. In database approach each primary fact is generally recorded in only one place in the database [6]. Sometimes it is desirable to include some limited redundancy to improve the performance of queries when it is more efficient to retrieve data from a single file instead of searching and collecting data from several files, but this data duplication is controlled by DBMS to prohibit inconsistencies among files. By eliminating data redundancy inconsistencies among data are also reduced [6]. Reducing reduncancy improves the consistency of data while reducing the waste in storage space. DBMS gives the opportunity of data sharing to the users. Sharing data often permits new data processing applications to be developed without having to create new data files. In general, less redundancy and greater sharing lead to less confusion between organizational units and less time spent resolving errors and inconsistencies in reports. The database approach also permits security restrictions. In a DBMS different types of authorizations are accepted in order to regulate which parts of the database various users can access or update.

2.2 Hệ thống dữ liệu

In contemporary computing environments, the dominance has shifted towards data-intensive applications, deviating from the traditional emphasis on compute-intensive tasks. The limiting factor for these applications seldom resides in the sheer computational power of the CPU; rather, the primary challenges typically revolve around the magnitude of the data, its intricate structures, and the rapidity with which it changes. Unlike compute-intensive operations that heavily rely on processing speed, data-intensive applications, dealing with extensive datasets, intricate data structures, or swiftly evolving information, necessitate adept strategies for storage, retrieval, and manipulation. Consequently, effectively addressing the multifaceted dynamics of data becomes paramount, highlighting the imperative for sophisticated data management and processing techniques to optimize performance in the face of these intricate challenges.

Why should we amalgamate these diverse elements within the overarching label of data systems? Recent years have witnessed the emergence of a plethora of novel tools for data storage and processing, each meticulously optimized for an array of distinct use cases [11]. Consider, for instance, datastores that concurrently function as message queues (e.g., Redis) or message queues equipped with database-like durability assurances (such as Apache Kafka). The demarcation lines between these categories are progressively fading, reflecting a landscape where boundaries are increasingly ambiguous.

Moreover, a growing number of applications now present challenges of such magnitude or diversity that a solitary tool is no longer sufficient to fulfill all its data processing and storage requisites. Instead, the workload is deconstructed into tasks amenable to efficient execution by individual tools. These disparate tools are then intricately interwoven using application code, offering a nuanced and adaptable approach to the multifaceted demands of contemporary data management and processing.

In navigating the complex landscape of application development, the quest for reliability, scalability, and maintainability unveils a challenging yet essential journey. As we delve into the intricate patterns and techniques that permeate various applications, we embark on an exploration to fortify these foundational pillars in the realm of software systems.

Ensuring reliability involves the meticulous task of ensuring systems function correctly, even when confronted with faults. These faults may manifest in the hardware domain as random and uncorrelated issues, in software as systematic bugs that are challenging to address, and inevitably in humans who occasionally err. Employing fault-tolerance techniques becomes imperative to shield end users from specific fault types.

Scalability, on the other hand, necessitates the formulation of strategies to maintain optimal performance, particularly when facing heightened loads. To delve into scalability, it becomes essential to establish quantitative methods for describing load and performance. An illustrative example is Twitter's home timelines, which serve as a depiction of load, and response time percentiles providing a metric for measuring performance. In a scalable system, the ability to augment processing capacity becomes pivotal to sustaining reliability amidst elevated loads.

The facet-rich concept of maintainability essentially revolves around enhancing the working experience for engineering and operations teams interacting with the system. Thoughtfully crafted abstractions play a crucial role in mitigating complexity, rendering the system more adaptable and modifiable for emerging use cases. Effective operability, characterized by comprehensive insights into the system's health and adept management methods, also contributes to maintainability.

Regrettably, there exists no panacea for achieving instant reliability, scalability, or maintainability in applications. Nonetheless, discernible patterns and recurring techniques emerge across diverse application types, offering valuable insights into enhancing these critical attributes.

Data Transformation:

Data Transformation is a vital part of the data flow process, where data changes to meet specific requirements of the process or system. There are two main directions for implementing data transformation: batch processing and real-time processing.

In batch processing mode, data is processed in batches, often scheduled for processing at predefined intervals. This is suitable for tasks requiring the processing of large and complex datasets without an immediate response.

On the contrary, real-time processing is the method of processing data as soon as it arrives, without waiting for a large amount of data to accumulate. This is often preferred in applications demanding low latency, such as real-time event processing.

ETL:

ETL, an acronym for Extract, Transform, Load, stands as a fundamental methodology indispensable for orchestrating the seamless movement of data within storage ecosystems. This three-step process plays a pivotal role in shaping the lifecycle of data.

Firstly, in the "Extract" phase, data is sourced from diverse origins, ranging from databases to files and online services. This initial step lays the groundwork by retrieving relevant information from the varied reservoirs of data.

Subsequently, in the "Transform" phase, the extracted data undergoes a metamorphosis to align with the specific requirements of the target system. This transformative stage encompasses tasks such as data cleansing, format conversions, and even the computation of novel indices, ensuring the data is refined and tailored to suit its intended purpose.

Finally, the "Load" phase marks the culmination of the ETL process. At this juncture, the meticulously transformed data finds its destination, being loaded into the designated storage system. This storage system typically takes the form of a data warehouse or data lake, serving as the repository for the refined and purpose-adapted data. In essence, ETL encapsulates a systematic and indispensable approach to managing the intricate journey of data within the expansive realm of storage systems.

Data Pipe:

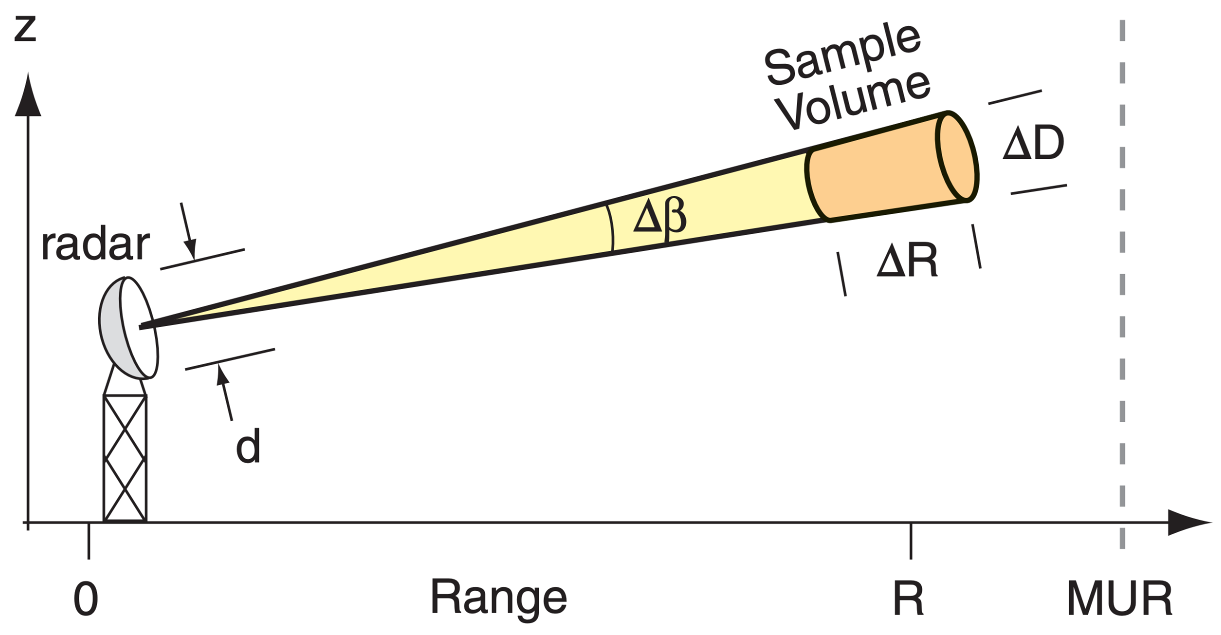
A data pipe is an essential concept in deploying effective data flow. Built on the idea of automating the movement and transformation of data, data pipes serve as powerful workflow streams.

Through the use of data pipes, large data volumes can be processed flexibly and efficiently. Tasks such as error handling, performance monitoring, and even deploying new transformations can be automated, minimizing manual intervention and enhancing system stability.

Data Orchestration:

Data Orchestration is the intricate process of coordinating and managing multiple data processes, workflows, or services to achieve specific outcomes. At its core, it involves the meticulous definition and management of workflows, determining the sequential order, and dependencies among various data processes. Task execution is finely tuned through controllers that schedule and orchestrate tasks at optimal times and in a precise sequence, ensuring the desired outcomes are achieved. Controllers play a pivotal role in managing dependencies between tasks, orchestrating the execution of tasks only when their dependent tasks are successfully met. Robust orchestration systems provide comprehensive tools for monitoring workflow progress and logging pertinent information, offering insights to address and rectify any potential issues. Moreover, controllers optimize performance by strategically breaking down complex tasks into multiple subtasks, executing them in parallel to enhance overall system efficiency.

2.3 Lý thuyết về khí tượng

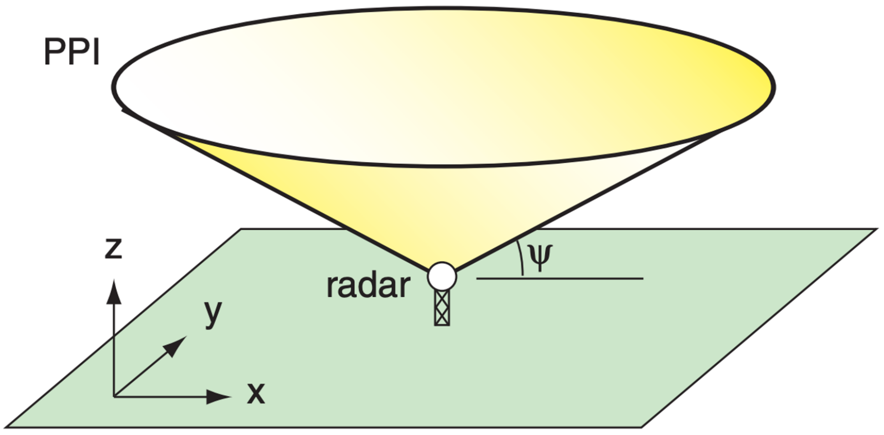


Hình 2.1: *Hệ thống radar thời tiết - [12]*

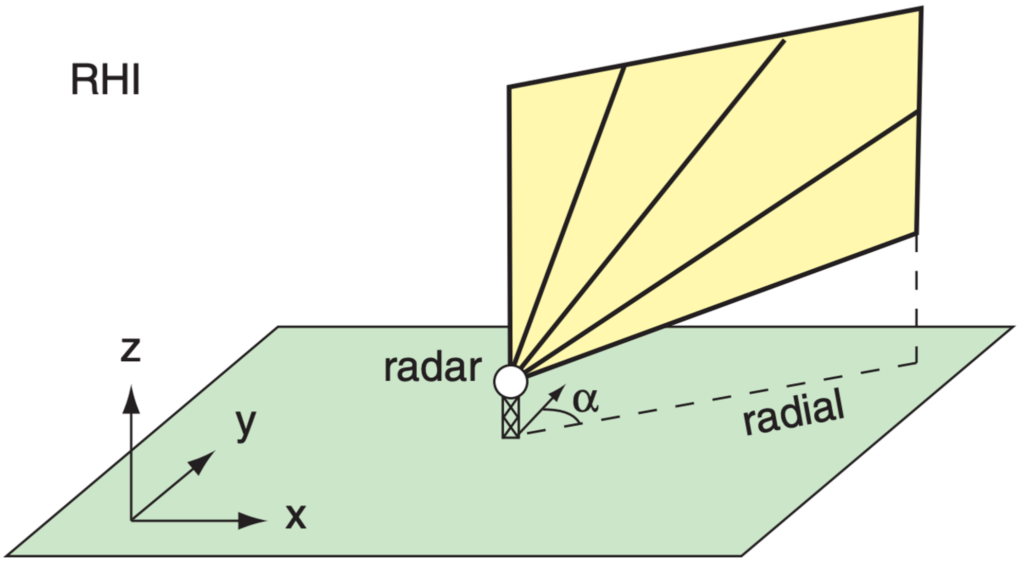
2.3.1 Basic terminologies

2.3.1.1 Weather Radar

Normally, weather radars are programmed to scan in an azimuth of 360^o. For every round the radar will scan at a different altitude. It usually takes about four to ten minutes for the radar to complete a full scan.



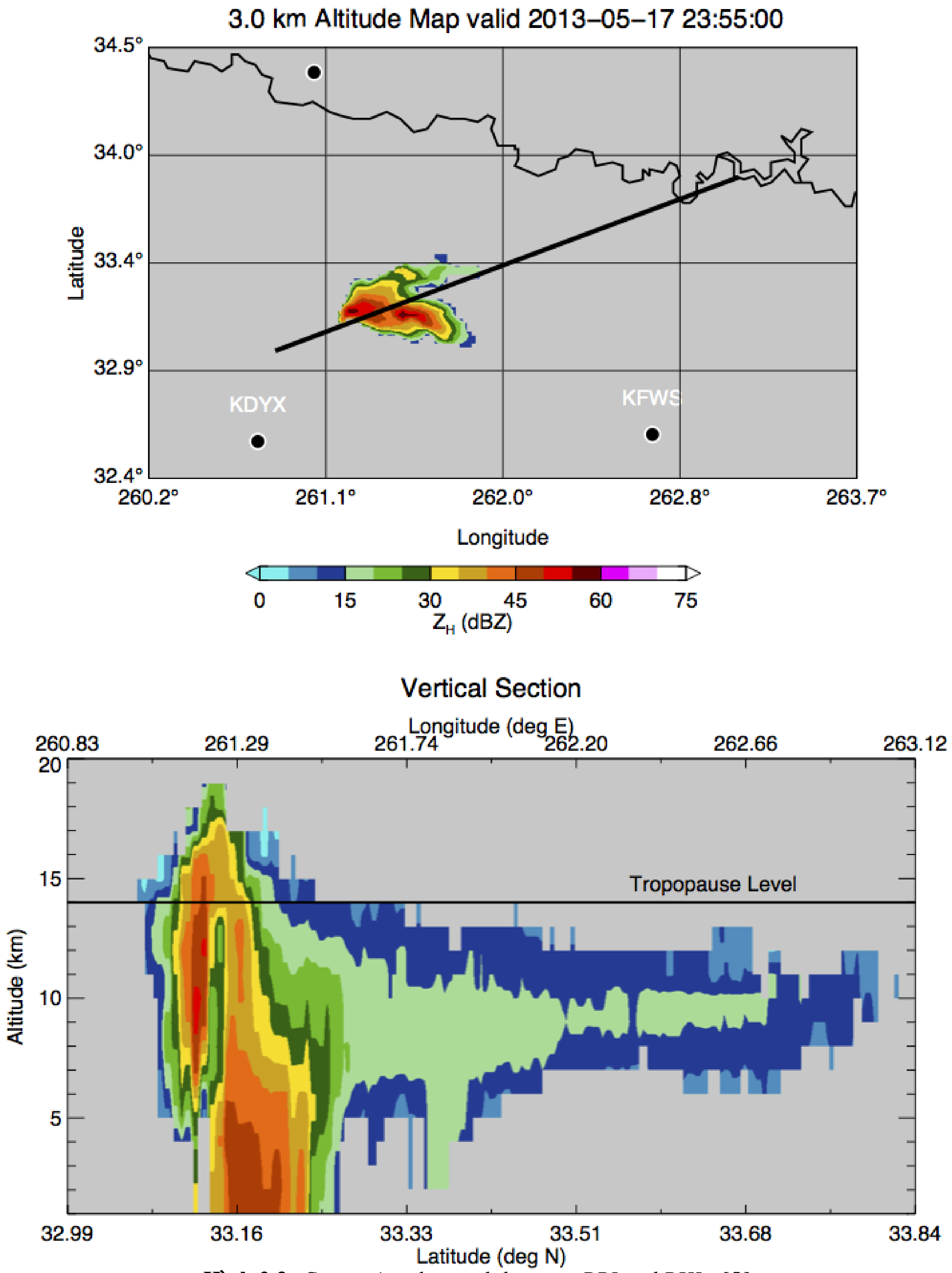
(a) *Plan-Position Indicator - PPI - [12]*



(b) *Range Height Indicator - [12]*

For PPI representation, the radar will scan the entire azimuth, but only at a certain altitude. The final result would be similar to a map on a flat surface.

For RHI, in contrast, the radar retains the azimuth but increases in altitude. The collected result gives viewers more details about the height and sizes of a meteorologist event.



Hình 2.3: *Comparing the result between PPI and RHI - [5]*

2.3.1.2 Radar equation and Reflectivity

At a certain point in time, weather radar will emit a short purse of radio wave ($\Delta t = 0.5 - 10\mu s$). Depending on the density of free molecules in the air (water vapor, smoke, ...), the energy of this wavelength will be partially absorbed. The wavelength intensity that the radar receives will be less than the intensity of the original wave. This ratio is expressed through **The radar equation** [12]:

$$\left[\frac{P_R}{P_T}\right] = [b] \cdot \left[\frac{|K|}{L_a}\right]^2 \cdot \left[\frac{R_1}{R}\right]^2 \cdot \left[\frac{Z}{Z_1}\right]$$

In which, the variables of the equation include:

- $|K|$ unitless:
  - $|K|^2 \approx 0.93$ for droplets
  - $|K|^2 \approx 0.208$ for ice crystal
- $R(\mathrm{km})$: distance from the radar to the target
- $R_1 = \sqrt{Z_1 \cdot c \cdot \Delta t / \lambda^2}$: ratio of distance
- Z: Radar's reflectivity
- $Z_1 = 1\ \mathrm{mm}^6\ \mathrm{m}^{-3}$: Radar's unit reflectivity

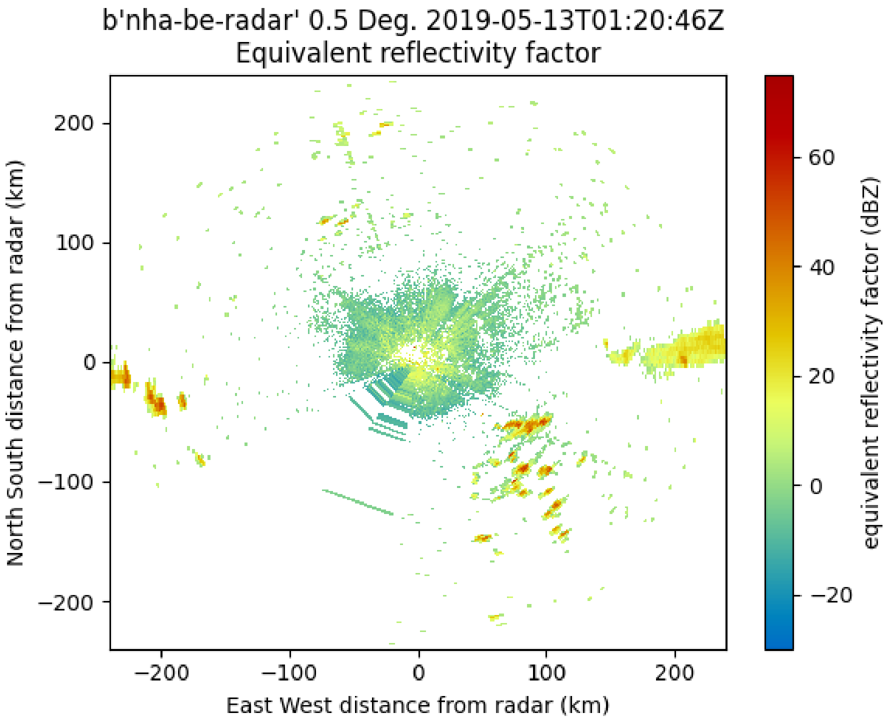
From the radar equation, we can derive the formula for reflectivity:

$$\mathrm{dBZ} = 10\left[\log\left(\frac{P_R}{P_T}\right) + 2\log\left(\frac{R}{R_1}\right) - 2\log\left|\frac{K}{L_a}\right| - \log(b)\right]$$

Meteorologists are usually interested in this number because it is proportional to the amount of precipitation.

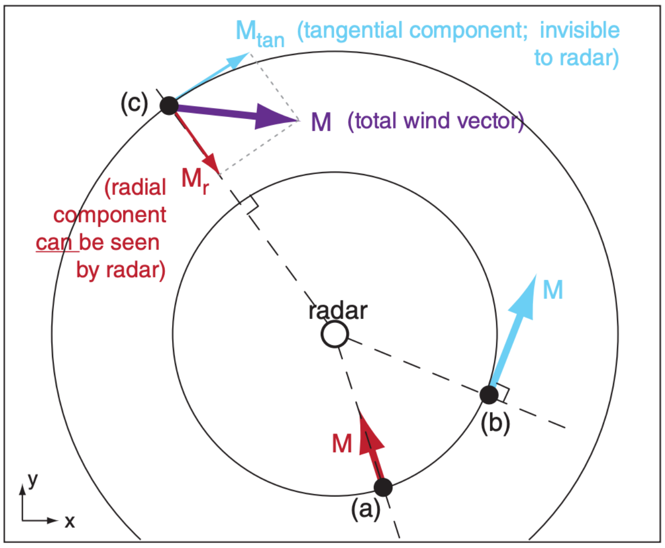
Value (dBZ)	Weather
-28	Haze
-12	Clear air
25 - 30	Dry snow / light rain
40 - 50	Heavy rain
75	Giant hail

Bảng 2.1: *Relation between reflectivity and precipitation - Stull [12]*



Hình 2.4: *Reflectivity from Nhà Bè radar*

2.3.1.3 Radial velocity


M_{tan} (tangential component; invisible to radar)
(c)
M (total wind vector)
M_r
(radial component can be seen by radar)
radar
M
(b)
M
(a)
y
x

Hình 2.5: *Illustrate the velocity situations that a Doppler radar can observe. (a) When the wind direction at point M coincides with the radius of the circle centered at the radar, the radar can determine the velocity at this point. (b) When the wind direction is tangent to the circle, the radar cannot determine the velocity. (c) Analyzing the wind direction at M into two perpendicular velocities, the radar can only determine the velocity vector along M_r.*

When the radio waves from these Doppler radars propagate to the molecules in the air, the displacement of these particles causes a phase shift between the transmitted and received signals. Radars rely on this information to calculate the wind velocity at various points in space.

2.4 Định dạng dữ liệu trong phân tích khí tượng

2.4.1 SIGMET data format - raw format (Vaisala)

Vaisala is a Finnish company specializing in environmental and meteorological instrumentation. The RAW format (also referred to as SIGMET in some documents [3]) is one of the storage formats developed by the company to organize data output from their radar devices.

Some notable points about this format include:

- The file content is divided into a **block**, each with a size of exactly 6144 bytes. This size aligns with the main storage size on older tape devices.
- The file typically consolidates data from all radar scanning sessions.

– Data records are organized within the scope of one block (6144 bytes). In the case of any remaining space in the block, it is padded with additional zeros.

With the aforementioned characteristics, the key advantages of the RAW format storage can be identified: [13]

– Compatibility with various tape types, which were commonly used devices in the past and are still widely used due to their cost-effective storage capacity.
– Through the use of the block mechanism, SIGMET facilitates block-level error recovery in storage systems.

The main concern raised by the team is the mapping capability between the storage structure on the hard drive and on the tape.

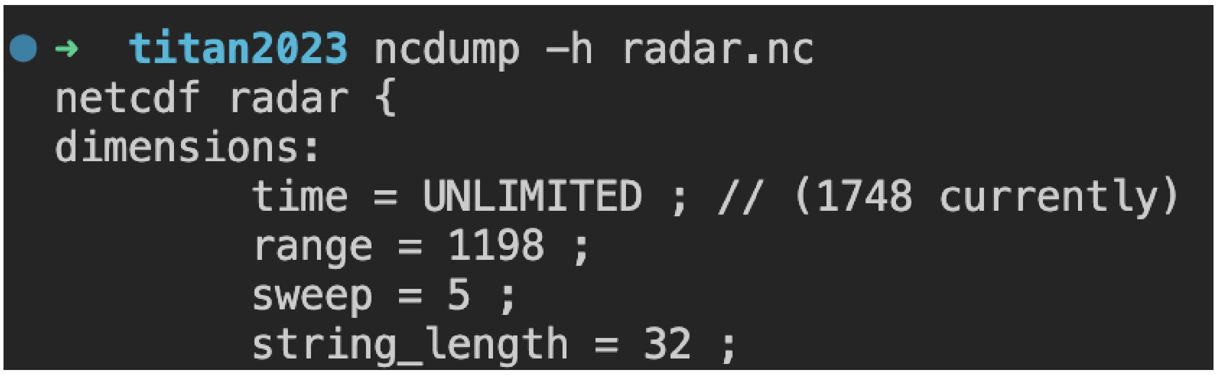
2.4.2 NETCDF data format - Network Common Data Form

NetCDF (Network Common Data Form) is a versatile file format designed explicitly for storing multidimensional scientific data. Within the netCDF library system, various binary formats are supported, each contributing to the flexibility and scalability of data management [10]. Notably, these formats include:

1. Classic Format: Initially used in the first version of netCDF and remains the default choice for file creation.
2. 64-bit Offset Format: Introduced since version 3.6.0, this format supports larger variable and file sizes.
3. netCDF-4/HDF5 Format: Introduced in version 4.0, utilizing the HDF5 data format with some limitations.
4. HDF4 SD Format: Primarily supports data reading.
5. CDF5 Format: Synchronized support with the parallel-NetCDF project.

All these formats exhibit self-description, with a detailed header part describing the file structure, including data arrays and file metadata in the form of attribute name/value pairs. This design ensures platform independence, with issues such as endianness being flexibly addressed through software libraries.

Consider a specific example of storing essential meteorological parameters such as temperature, humidity, pressure, wind speed, and direction in netCDF files. This illustrates the capability of this format in handling diverse scientific datasets, providing a powerful and flexible means to manage multidimensional information.



```
● ➜  titan2023 ncdump -h radar.nc
  netcdf radar {
  dimensions:
          time = UNLIMITED ; // (1748 currently)
          range = 1198 ;
          sweep = 5 ;
          string_length = 32 ;
```

Hình 2.6: *Radar information in NETCDF format. The total dimensions of the dataset are 2975, grouped into 4 distinct labels.*

Starting from version 4.0, the netCDF API introduces the ability to use the HDF5 data format. This crucial integration allows netCDF users to create HDF5 files, unlocking benefits such as significantly larger file sizes and support for unlimited dimensions. This step marks a significant move towards leveraging the extended advantages of the HDF5 format.

NetCDF Classic and 64-bit Offset Formats are international standards of the Open Geospatial Consortium [4], demonstrating the robustness and reliability in ensuring the compatibility and global scalability of the netCDF format.

2.5 Công nghệ sử dụng

2.5.1 Apache Airflow™

Apache Airflow™ stands as an open-source platform designed to manage data flow within systems associated with data. In the face of the escalating challenge of data pipeline management, Airflow emerges as a comprehensive solution, automating and optimizing data-related workflows effectively [1].

Airflow not only aids in defining and managing the start and end times of each data pipeline but also provides precise and detailed monitoring of the results of each task. This becomes particularly crucial when ensuring the integrity and reliability of the processed data.

With the ability to discern complex relationships between tasks through the Directed Acyclic Graph (DAG) model, Airflow empowers administrators with tighter control and flexibility in handling workflow processes. Its robust integration with logging systems facilitates detailed activity tracking, assisting in issue resolution and ensuring that every process aligns with expectations.

Simultaneously, the scheduling flexibility makes Airflow an excellent tool for time and resource management. Its strong integration with various data sources and extensibility through plugins allows Airflow to meet diverse needs in data processing and task automation.

Apache Airflow not only delivers robust performance but also brings flexibility and optimal technical features to data processing workflows. With its time management capabilities, powerful logging integration, scheduling flexibility, and scalability, Airflow stands as the top choice for enhancing performance and control in data processing workflows.

2.5.2 Kubernetes

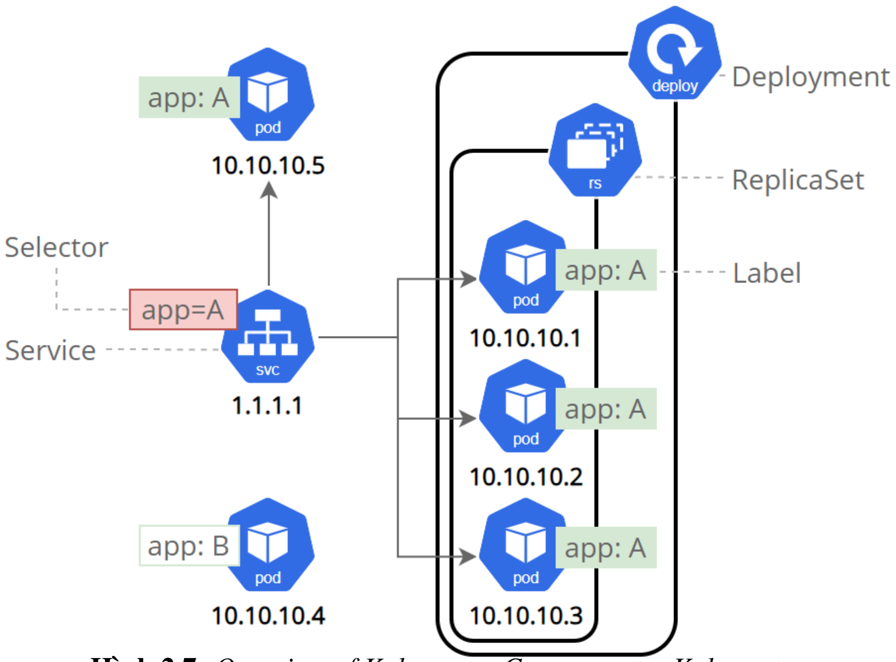
Kubernetes, an open-source system for managing and deploying highly flexible applications in cloud and data center environments, has evolved into one of the most widely adopted tools in the field of Information Technology [2]. Originally developed by Google and later transferred to the Cloud Native Computing Foundation (CNCF), Kubernetes aims to automate the deployment, scaling, and management of containerized applications, alleviating the burden on developers and system administrators. The platform offers a unified foundation for deploying, scaling, and managing containerized applications across multiple servers.

Kubernetes operates based on key concepts such as Pods, Services, ReplicaSets, and various other abstractions, creating a flexible environment for application deployment and management. This fosters an environment where developers can easily build applications, and system administrators can efficiently maintain them.

Beyond supporting traditional deployment models, Kubernetes paves the way for innovative strategies like Continuous Deployment (CD) and Microservices. With the ability to automate many aspects of the development and deployment process, Kubernetes plays a crucial role in constructing and sustaining complex, flexible, and scalable systems.

2.5.2.1 Kubernetes Components

Introducing essential concepts for managing and deploying applications, Kubernetes provides an effective and flexible environment. The main components of Kubernetes include Pod, ReplicaSet, Deployment, and Service.



Hình 2.7: *Overview of Kubernetes Components - Kubernetes*

In Kubernetes, a **Pod** serves as the fundamental unit, representing a collection of containers that share a common workspace. Within the same Pod, containers collaborate by sharing network and storage resources, fostering interaction and enabling the construction of intricate applications.

The **ReplicaSet**, a crucial resource in Kubernetes, ensures a designated number of Pods operate in a specified manner. In the event of a Pod failure or shutdown, the ReplicaSet automatically initiates the creation of a new Pod to replace it. This mechanism ensures the application's stable state by guaranteeing a defined number of Pods are consistently operational.

For managing the deployment and updating processes of applications, **Deployment** is a key component in Kubernetes. It articulates the desired state of the application and orchestrates the updating of the ReplicaSet to achieve that state. Deployment provides versatile management capabilities, facilitating the deployment of new versions, rollbacks, and updates without disrupting the service.

The **Service** resource in Kubernetes furnishes an HTTP port to Pods, generating a unique IP address and DNS name for a cluster of Pods. This enables seamless communication among applications within the cluster and with external environments. Service effectively simplifies the intricacies of handling multiple Pods and IP addresses, offering a straightforward means of accessing services within the Kubernetes environment.

Typically, large-scale systems leverage Kubernetes in their software development and deployment processes. This adoption brings several advantages, including efficient resource management and self-recovery capabilities. Kubernetes optimizes resource utilization, ensuring optimal performance and reducing waste. Additionally, it automatically addresses issues during operations, enhancing high availability.

However, the technology is not without its challenges, including a steep learning curve for beginners. Mastery of diverse knowledge areas such as computer networking and containerization is necessary. Moreover, deploying and maintaining Kubernetes demands significant resources, both in terms of personnel and hardware, particularly for smaller organizations.

2.5.2.2 High Availability in Kubernetes

High Availability is a crucial factor in the success of any system. In Kubernetes, High Availability is achieved through the combination of several features, including self-healing, load balancing, and auto-scaling.

First, Kubernetes uses **Deployment**, a type of **Controller** for managing replicas. Through Deployment, we can easily perform horizontal scaling, which is the process of increasing the number of replicas of a Pod. This mechanism ensures that the application can handle numerous requests without compromising performance. Moreover, in case of a Pod failure, a Deployment makes sure that a new Pod is created to replace it, ensuring the amounts of predefined replicas is always maintained.

At network layers, Kubernetes uses **Service** for communication between various components within or outside the cluster. Instead of directly accessing Pods, other components can access Services, which will redirect the request to the appropriate Pod. When Pods are replaced, the Service will automatically update the routing rules to ensure the request is sent to the correct Pod. Outside the cluster, Kubernetes also uses **Ingress** to manage external access to Services. Instead of specifying the direct node IP address, Clients can abstract it by using only the URL or hostname.

Finally, at the storage level, Kubernetes provides Persistent Volumes. By doing so, applications can be agnostic to the underlying storage infrastructure. This allows for easier management and scaling of storage resources. Not only that, depending on the provided **Storage Class**, Kubernetes makes sure that the data is replicated to multiple nodes, ensuring data availability in case of node failure.

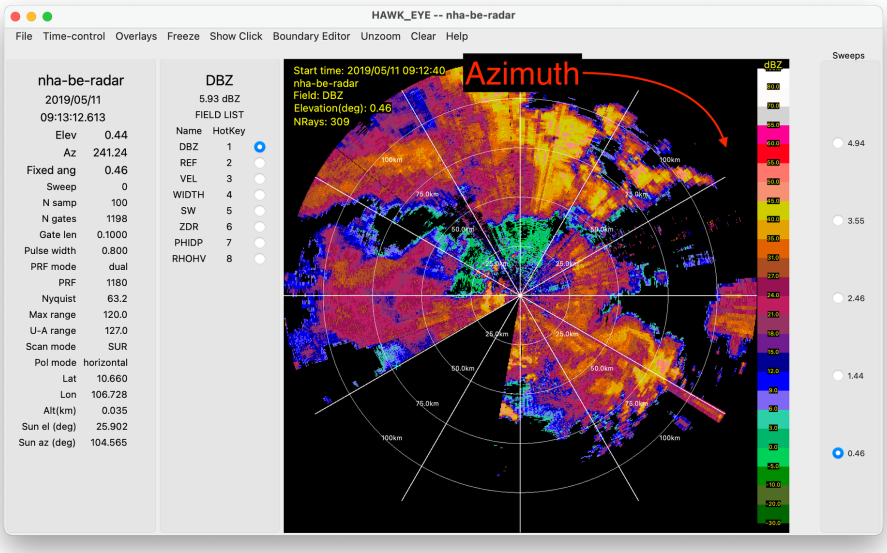
To summarize, Kubernetes provides a robust set of features to ensure High Availability. By leveraging these features, we can build a highly available system that can scale with our load, while also being resilient to failures.

2.5.3 LROSE

LROSE (Lidar Radar Open Software Environment) is a project supported by the National Science Foundation (NSF) with the goal of developing common software for the Lidar, Radar, and Profiler community. The project operates based on the principles of collaboration and open source. The core software package of LROSE is a collaborative effort between Colorado State University (CSU) and the Earth Observing Laboratory (EOL) at the National Center for Atmospheric Research (NCAR) [8].

Originating from the need for a unified software environment for processing Lidar and Radar data in atmospheric science research [8], the project addresses complexities related to integrating data from various observation platforms, including Lidar, Radar, and Profiler. These components are designed to meet the specific needs of meteorologists and researchers working with remote sensor data.

LROSE is widely used in meteorological research, including studies related to cloud and precipitation processes, boundary layer dynamics, and other meteorological phenomena. The software supports the analysis of observation data collected from ground-based tools such as Lidar and Radar. LROSE seamlessly integrates with a variety of model and atmospheric analysis tools to optimize its capabilities. Researchers often integrate LROSE into numerical weather prediction models as well as other data assimilation techniques, creating a flexible and powerful system.


HAWK_EYE -- nha-be-radar
File Time-control Overlays Freeze Show Click Boundary Editor Unzoom Clear Help
Azimuth
dBZ
Sweeps

nha-be-radar
2019/05/11
09:13:12.613

Elev	0.44
Az	241.24
Fixed ang	0.46
Sweep	0
N samp	100
N gates	1198
Gate len	0.1000
Pulse width	0.800
PRF mode	dual
PRF	1180
Nyquist	63.2
Max range	120.0
U-A range	127.0
Scan mode	SUR
Pol mode	horizontal
Lat	10.660
Lon	106.728
Alt(km)	0.035
Sun el (deg)	25.902
Sun az (deg)	104.565

DBZ
5.93 dBZ
FIELD LIST

Name	HotKey	
DBZ	1	●
REF	2	○
VEL	3	○
WIDTH	4	○
SW	5	○
ZDR	6	○
PHIDP	7	○
RHOHV	8	○

Start time: 2019/05/11 09:12:40
nha-be-radar
Field: DBZ
Elevation(deg): 0.46
NRays: 309

Sweeps: 4.94, 3.55, 2.46, 1.44, [✓ 0.46]

Hình 2.8: *Hawk Eye, Lidar and Radar visualization tool of LROSE*

The project actively encourages participation from a large scientific community, promoting the exchange of ideas, algorithms, and improvements for the software. Regular updates and contributions from users contribute to the continuous development and refinement of LROSE.

2.5.4 FastAPI

FastAPI is a high-performance web development framework based on Python, distinguished by its unique and advanced technical features designed to address challenges in building high-performance and flexible APIs.

It is one of the rare frameworks that fully leverages type hints and async/await. Type hints help clearly define the data types of variables and functions, making the source code more explicit and understandable. Additionally, support for async/await enables handling multiple requests concurrently without slowing down the processing, especially crucial for applications requiring high responsiveness and scalability.

With robust support for type hints, FastAPI automatically generates API documentation based on Python's data types. This not only helps reduce errors in the source code but also generates automatic API documentation, streamlining the development process and interaction with the API. FastAPI also supports industry standards such as OpenAPI and Swagger, providing a flexible way to manage resources, authentication, and user accounts.

Beyond its technical prowess, FastAPI excels in concurrent handling and performance. The special support from asyncio allows FastAPI to process multiple requests simultaneously without impeding processing speed. This capability ensures smooth and fast-running applications while facilitating scalability, a crucial factor for large research projects with evolving requirements over time.

FastAPI not only simplifies the development process but is also designed with the goal of optimizing the development experience. It lays the foundation for easily building production-ready APIs thanks to its integrated best practices. With this combination, FastAPI is not just a powerful tool but also an efficient and flexible solution for projects that demand reliability and scalability in the future.

Chương 3

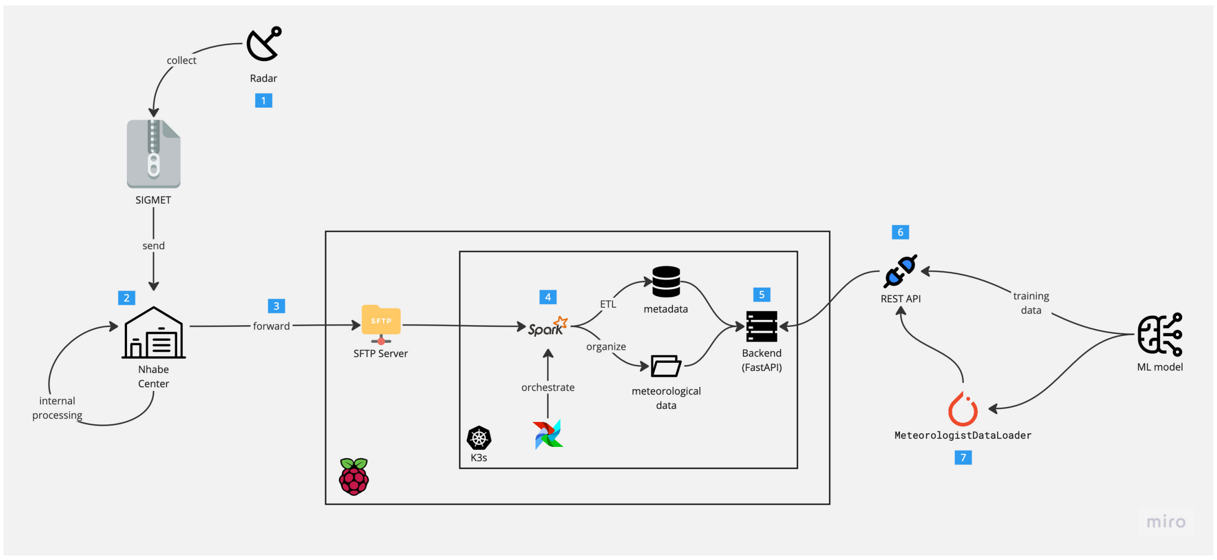
Phân tích và thiết kế hệ thống

3.1 Tổng quan

3.2 Thăm dò dữ liệu

3.3 Mô hình cơ sở dữ liệu

3.4 Kiến trúc hệ thống



Hình 3.1: *Thiết kế hệ thống - đề xuất*

Dựa vào những yêu cầu đặt ra từ các bên liên quan, cũng như sau khi được tìm hiểu về hệ thống hiện tại, nhóm đề xuất thiết kế và hiện thực một hệ thống "Cơ sở dữ liệu tích hợp cho hệ thống dự đoán ngắn hạn trong Khí tượng thuỷ văn". Hình 3.1 minh hoạ cho mẫu thiết kế của nhóm.

3.5 Hệ thống hiện tại

Nội dung thiết kế gồm 7 phần chính. Trong đó, phần hệ thống đang trong quá trình hoạt động tại trạm quan trắc Nhà Bè được thể hiện ở hai bước đầu tiên:

Trước tiên, tại bước 1, trong mỗi chu kỳ nhất định, vệ tinh địa tĩnh sẽ thu nhập dữ liệu về khí tượng. Một số những chỉ số thu nhận được có thể kể đến như:

1. Độ rộng phổ Doppler (Doppler spectrum width)
2. Tốc độ gió Doppler trung bình (Mean doppler velocity)
3. Phản hồi vô tuyến (Reflectivity)

Tất cả những thông tin này sẽ được truyền về trạm quan trắc tại Nhà Bè dưới định dạng SIGMET 2.4.1. Sau đó, ở bước 2, những nhân viên tại trạm khí tượng thuỷ văn Nhà Bè sẽ tiến hành ghi nhận và xử lý các dữ liệu được truyền về, tuỳ theo các yêu cầu về nghiệp vụ của họ.

Chương 4

Hiện thực

4.1 Luồng dữ liệu

Phần hiện thực của nhóm sẽ nằm trong năm bước còn lại trong mô tả tại hình 3.1.

Tại bước 3, nhóm sẽ setup (cài đặt) một server SFTP đơn giản. SFTP là một giao thức đơn giản và phổ biến. Hiện nay, có rất nhiều những thư viện và công cụ để hỗ trợ giao tiếp dựa trên giao thức này. Ngoài ra, so với FTP, giao thức kể trên còn đảm bảo tính bảo mật trong suốt quá trình chuyển dịch dữ liệu. Tuỳ thuộc vào mức độ cho phép, nhóm có thể hỗ trợ phía trạm quan trắc xây dựng các scripts (kịch bản) để tự động forward (chuyển tiếp) các file sau khi đã được xử lý tại đây. Hoặc ngược lại, phía trạm quan sát có thể gửi file đến server trên một cách thủ công.

Khi file đã được upload đến server SFTP, nhóm sử dụng Airflow để điều hành tất cả các luồng ETL hiện có trong hệ thống chung. Ở thời điểm hiện tại, nhóm chỉ dừng lại với một DAG duy nhất, để xử lí dữ liệu đến từ trạm quan trắc Nhà Bè. Airflow sẽ tiến hành quan sát những file được thêm mới vào server SFTP của chúng ta và khởi chạy ETL. Việc lựa chọn Apache Spark ở đây dựa trên khối lượng dữ liệu và độ phức tạp được đặt ra. Nếu lượng dữ liệu là không quá nhiều cho mỗi file SIGMET mới, và bản thân Python có xử lí được, không cần thiết phải sử dụng Spark ở bước này.

Dữ liệu về khí tượng khi được đưa đến cơ sở hạ tầng của nhóm sẽ được phân ra hai luồng chính: Những metadata (thông tin mở rộng) của dữ liệu gốc như ngày tạo ra, kích thước, thời điểm ghi nhận, ... sẽ được lưu trong một RDBMS (hệ quản trị cơ sở dữ liệu quan hệ) truyền thống. Cụ thể ở đây, nhóm lựa chọn PostgreSQL nhờ vào độ phổ biến và mức độ am hiểu của nhóm. Các metadata lưu trữ ở đây giúp cơ sở dữ liệu của nhóm nhanh chóng phản hồi các query (truy vấn) mà chưa cần trực tiếp phải sử dụng đến dữ liệu gốc. Một số query phổ biến có thể kể đến như:

- Các mốc thời gian đang được ghi nhận bao gồm những gì? (Ví dụ: từ ngày 21/11/2023 cho đến ngày 17/12/2023)
- Tại thời điểm x, radar có toạ độ địa lý là bao nhiêu?
- Các trường dữ liệu đang được lưu trữ là gì?

Bên cạnh đó, DB (cơ sở dữ liệu) trên còn đóng vai trò như mục index (chỉ mục) giúp hệ thống nhanh chóng xác định vị trí lưu trữ của dữ liệu gốc.

Với các dữ liệu về khí tượng thuỷ văn cụ thể, nhóm nhận thấy rằng sẽ không thật sự hiệu quả khi lưu trữ chúng trực tiếp trên các DBMS trên. Đồng thời, nhóm nhận thấy việc lưu trữ dữ liệu trên file vẫn đem đến một kích thước tổng quan hợp lý. Vì vậy, nhóm quyết định sẽ tách phần dữ liệu thô ra và lưu trữ trực tiếp trên các files. Đồng thời kết hợp với các index (đã đề cập ở trên) để tăng tốc quá trình truy xuất.

Để tạo cửa ngõ cho việc truy vấn dữ liệu, phục vụ cho các bên về model, machine learning và AI, ... nhóm sẽ phát triển một server Backend đơn giản, sử dụng FastAPI của Python để giúp tăng tốc độ phát triển giải pháp. Tại bước 5, backend sẽ nhận dữ liệu truy vấn dưới định dạng REST API (tại bước 6), truy vấn dữ liệu trong DB của metadata và trong các file dữ liệu và trả về kết quả đạt được. Ở những lần train khác nhau, các bên của Machine Learning có thể kết nối đến server này để lấy dữ liệu.

Cần nói thêm, toàn bộ hệ thống sẽ được phát triển và vận hành theo hướng containerize (đóng gói) và sẽ được deploy (triển khai) trên nền tảng Kubernetes. Việc này thể hiện khả năng của hệ thống trong việc duy trì tính sẵn sàng cao (High-Availability) cũng như dễ dàng trong việc duy trì giải pháp. Trong phạm vi phần minh hoạ này, nhóm sẽ chỉ dừng lại với việc triển khai trên một cụm máy tính nhúng Raspberry Pi.

Sau cùng, tại bước 7, nhóm muốn đề xuất thêm một vấn đề. Nếu phù hợp, nhóm có thể xây dựng thêm một `DataLoader` (bộ nạp dữ liệu) để phục vụ nhanh chóng đến các nhóm làm model khác. Một trong những thư viện phổ biến hiện nay của các bên AI là Pytorch, nên nhóm sẽ tiếp cận với nền tảng này trước.

Chương 5

Kiểm thử

5.1 Unit testing

5.2 Integrated testing

Trong nghiên cứu này, chúng tôi đã thành công trong việc xây dựng một Proof-of-Concept (chứng minh khái niệm) mang tính ứng dụng cao, nhằm mục đích giảm thiểu các công đoạn trong quy trình làm việc thông thường. Đây là một bước quan trọng để tối ưu hóa và cải thiện hiệu suất làm việc trong các ngữ cảnh nghiên cứu và thực tế.

Chúng tôi đã đặt ra mục tiêu tạo ra một hệ thống linh hoạt có khả năng thích ứng cao, giúp giảm bớt những bước phức tạp trong quy trình công việc. Bằng cách này, chúng tôi không chỉ giúp tăng cường hiệu suất mà còn giảm áp lực công việc đối với nhân sự, tạo điều kiện thuận lợi cho sự sáng tạo và tập trung vào các nhiệm vụ chính.

Chúng tôi không chỉ dừng lại ở việc phát triển hệ thống mà còn đề xuất các chiến lược triển khai linh hoạt, nhấn mạnh sự tích hợp dễ dàng vào môi trường làm việc hiện tại của những người đang thực hiện công việc thu thập thông tin và dự báo thời tiết.

Chương 6

Hướng phát triển

Phát triển thành Hệ thống nền tảng dữ liệu thời tiết

Hệ thống nền tảng dữ liệu thời tiết (Weather Data Platform - WDP) được phát triển với mục tiêu trở thành một giải pháp toàn diện cho việc khai thác sức mạnh của dữ liệu thời tiết. Hệ thống được thiết kế để đáp ứng những yêu cầu cụ thể của các nhà khí tượng, học giả, nghiên cứu học thuật và các nhà phát triển từ nhiều lĩnh vực khác nhau, bao gồm cả freelancers, doanh nghiệp và tổ chức phi chính phủ (Non-governmental Organizations - NGOs). WDP đóng vai trò như một trung tâm tập trung cho việc tích hợp dữ liệu thời tiết, phân tích, và nhiều tính năng khác.

Với mong muốn phát triển thành Hệ thống nền tảng dữ liệu thời tiết, chúng tôi hướng đến sự hoàn thiện và đa chiều hoá thông tin thời tiết. Không chỉ là một bảng số liệu, mà là một trải nghiệm toàn diện. Trong tương lai, bên cạnh việc tiếp tục xây dựng cơ sở dữ liệu tích hợp theo hướng đã đề xuất, chúng tôi hứa hẹn sẽ tiếp tục nghiên cứu để mở rộng và phát triển cơ sở dữ liệu tích hợp này thành hệ thống nền tảng dữ liệu thời tiết với những hướng phát triển như sau:

1. **Dữ liệu phi tuyến:** Mở rộng từ việc tích hợp dữ liệu cơ bản, chúng tôi sẽ chú trọng vào việc cung cấp dữ liệu phi tuyến, chi tiết và đa nguồn, giúp người dùng khám phá thêm về môi trường xung quanh.

2. **Trí tuệ nhân tạo thấu hiểu**: Sử dụng trí tuệ nhân tạo để thấu hiểu ngôn ngữ của thời tiết, từ những biến đổi nhỏ đến những sự kiện lớn, tạo nên một nguồn thông tin thời tiết sâu sắc và thông minh.

3. **Giao diện người dùng tương tác**: Không chỉ là việc truy cập thông tin, mà còn là việc tương tác với dự báo thời tiết. Giao diện người dùng sẽ là nơi người dùng thể hiện sự tò mò và tương tác trực tiếp với dữ liệu.

4. **Kết Nối Thông Tin Địa Lý**: Tận dụng hệ thống thông tin địa lý để mang đến cái nhìn thực tế hóa, địa bàn hóa cho dự báo thời tiết. Điều này giúp người dùng hiểu rõ hơn về tác động thời tiết đối với môi trường xung quanh họ.

5. **Tối ưu hoá hiệu suất**: đảm bảo khả năng đáp ứng nhanh chóng và đồng đều trong mọi điều kiện.

6. **Bảo mật dữ liệu**: Tăng cường an toàn dữ liệu để đảm bảo tính bảo mật và toàn vẹn của thông tin thời tiết.

7. **Hệ thống dự báo nâng cao**: Nghiên cứu và tích hợp trí tuệ nhân tạo để cải thiện khả năng dự báo và đưa ra thông tin dự báo cáo chính xác.

8. **Kiểm thử và tối ưu hoá**: Tiến hành kiểm thử hệ thống để đảm bảo tính ổn định và xử lý mọi vấn đề tiềm ẩn. Tối ưu hóa hiệu suất nếu cần.

9. **Triển khai và duy trì**: Triển khai hệ thống và duy trì một chu kỳ cập nhật đều đặn để đảm bảo rằng nó luôn cung cấp thông tin thời tiết chính xác và đáng tin cậy.

Tài liệu tham khảo

[1] What is airflow? URL https://airflow.apache.org/docs/apache-airflow/stable/index.html. Truy cập lần cuối ngày 16/12/2023.

[2] Overview. URL https://kubernetes.io/docs/concepts/overview/. Truy cập lần cuối ngày 17/12/2023.

[3] Radxconvert - lrose wiki. URL http://wiki.lrose.net/index.php/RadxConvert.

[4] Ogc standard netcdf classic and 64-bit offset. https://www.opengeospatial.org/standards/netcdf, Accessed: 2017-12-05. Archived from the original on 2017-11-30. Retrieved 2017-12-05.

[5] casey. Using range height indicator scan of radar. URL https://earthscience.stackexchange.com/questions/7222/using-range-height-indicator-scan-of-radar. Truy cập lần cuối ngày 17/12/2023.

[6] Ramez A. Elmasri and Shamkant Navathe. *Fundamentals of Database Systems*. Addison Wesley, third edition, 1998.

[7] TRUNG TÂM DỰ BÁO KHÍ TƯỢNG THỦY VĂN QUỐC GIA, Jun 2203.

[8] Brenda Javornik, Hector Santiago III, and Jennifer C. DeHart. The lrose science gateway: One-stop shop for weather data, analysis, and expert advice. In *Practice and Experience in Advanced Research Computing*, PEARC '21. ACM, July 2021. doi: 10.1145/3437359.3465595. URL http://dx.doi.org/10.1145/3437359.3465595.

[9] G. Latisen and G. Vossen. *Models and Languages of Object-Oriented Databases*. Addison Wesley, 1998.

[10] Russ Rew, Glenn Davis, Steve Emmerson, Cathy Cormack, John Caron, Robert Pincus, Ed Hartnett, Dennis Heimbigner, Lynton Appel, and Ward Fisher. Unidata netcdf, 1989. URL `http://www.unidata.ucar.edu/software/netcdf/`.

[11] Michael Stonebraker and Uğur Çetintemel. 'one size fits all': An idea whose time has come and gone. In *Proceedings of the 21st International Conference on Data Engineering (ICDE)*, April 2005.

[12] Roland Stull. Weather Radars, 12 2022. [Truy cập lần cuối ngày 17/12/2023].

[13] *RAW Product Format - IRIS Programming Guide - IRIS Radar*. Vaisala. URL `https://ftp.sigmet.vaisala.com/files/html_docs/IRIS-Programming-Guide-Webhelp/raw_product_format.html`.