IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 3, December 2023

Overview of Apache NiFi

Sushma Laxman Wakchaure
Professor, Department of Computer Technology
Amrutvahini Polytechnic, Sangamner, Maharashtra, India

Abstract: *Apache NiFi is an open-source data integration tool designed to automate the flow of data between systems. It provides a user-friendly interface to design data flows and move data between different systems, making it easier to collect, distribute, and manage data in real-time. NiFi supports the concept of data flow, allowing users to define and visualize the movement of data from various sources to various destinations.At its core, Apache NiFi is a data integration tool that facilitates the automated flow of data between systems. It was developed by the National Security Agency (NSA) and later open-sourced as part of the Apache Software Foundation. NiFi is designed to be scalable, flexible, and easy to use, making it suitable for a variety of data integration scenarios.Providing a user-friendly interface for designing, managing, and monitoring data flows, NiFi facilitates the seamless movement of data across different environments. Utilizing a visual design paradigm, users can construct data flows by connecting a variety of processors that perform tasks such as data ingestion, transformation, and routing. NiFi's scalability, provenance tracking, and security features make it well-suited for handling complex data integration scenarios. With a robust community and ongoing development, Apache NiFi continues to play a crucial role in simplifying and optimizing data movement and processing workflows*

Keywords: NiFi

I. INTRODUCTION

1.1 What is Apache NiFi?

Put simply, NiFi was built to automate the flow of data between systems. While the term 'dataflow' is used in a variety of contexts, we use it here to mean the automated and managed flow of information between systems. This problem space has been around ever since enterprises had more than one system, where some of the systems created data and some of the systems consumed data. The problems and solution patterns that emerged have been discussed and articulated extensively. A comprehensive and readily consumed form is found in the Enterprise Integration Patterns. Framework for data processing that enables users to gather, transform, and transfer data from edge devices to cloud computing infrastructure. NiFi's fundamental design concepts closely relate to the main ideas of Flow Based Programming.

Difference between flow based programming and apache NI-FI-

Flow-based programming (FBP) is a programming paradigm that focuses on defining applications as networks of "black-box" components, called nodes, that communicate with each other by sending and receiving data through interconnected edges. Each node in the network is a self-contained module responsible for a specific function, and the data flows between these nodes to achieve the overall application functionality. Apache NiFi, on the other hand, is an open-source data integration tool that implements the principles of flow-based programming to automate the movement and transformation of data.

Here are some key differences and similarities between Flow-Based Programming and Apache NiFi:

Scope and Purpose:

- Flow-Based Programming: FBP is a general programming paradigm that can be applied to various domains, including software development, simulation, and data processing.
- Apache NiFi: NiFi is specifically designed for data integration, providing a visual interface to design, control, and monitor the flow of data between different systems.

Components:

- Flow-Based Programming: FBP systems consist of nodes and edges, where nodes represent processing components, and edges represent the flow of data between them.
- Apache NiFi: NiFi has similar concepts, with processors representing the processing nodes, connections defining the data flow (edges), and various other components such as Input and Output Ports, Controller Services, etc.

Use Cases:

- Flow-Based Programming: FBP is a general-purpose paradigm and can be applied to a wide range of applications, including non-data-centric scenarios.
- Apache NiFi: NiFi is specifically designed for data integration use cases, such as data ingestion, transformation, and movement in real-time.

Visual Representation:

- Flow-Based Programming: FBP systems often use visual tools to represent the flow of data, making it easier for users to design and understand the structure of the application.
- Apache NiFi: NiFi provides a visual web-based interface that allows users to design and manage data flows using a drag-and-drop approach, providing a visual representation of the data integration process.

Domain-Specific Features:

- Flow-Based Programming: FBP is a general paradigm and does not come with domain-specific features tailored for data integration.
- Apache NiFi: NiFi is specifically built for data integration, and it includes features such as data provenance tracking, content-based routing, and various processors for specific data handling tasks.

In summary, while Flow-Based Programming is a broader programming paradigm, Apache NiFi is a specific implementation of this paradigm tailored for data integration. NiFi provides a user-friendly interface and features designed to simplify the design, control, and monitoring of data flows in the context of data integration tasks.

Key concepts in Apache NiFi include:

1. Processor: Processors are the fundamental building blocks of a NiFi data flow. They perform the actual work of moving, transforming, and routing data. NiFi has a wide range of processors to handle different tasks, such as reading files, executing scripts, converting data formats, and interacting with external systems.
2. FlowFile: A FlowFile represents a single piece of data in NiFi. It encapsulates the data content and attributes associated with that data. FlowFiles are used to track and move data through the NiFi data flow.
3. Connection: Connections define the relationships between processors and indicate how data should flow from one processor to another. They specify the rules for data movement, transformation, and routing within the data flow.
4. Flow Controller: The Flow Controller is the central component that manages the overall execution of the data flow. It schedules and coordinates the activities of processors, ensuring that data flows smoothly through the system.
5. Input and Output Ports: Input and Output Ports define the entry and exit points of data into and out of a NiFi data flow. Ports are used to connect NiFi to external systems, allowing data to be ingested or sent to external destinations.
6. Controller Services: Controller Services are shared services that provide configuration settings and credentials to processors. They are used to centralize configuration information and promote reusability across multiple processors.

7. NiFi Registry: NiFi Registry is a complementary tool that works with Apache NiFi to provide version control and management of data flow configurations. It allows users to track changes, manage versions, and deploy data flows across different environments.
8. Template: A template is a reusable and shareable representation of a NiFi data flow. Users can create templates to encapsulate a set of processors, connections, and settings, making it easy to replicate and share data flow designs.

Apache NiFi is commonly used in scenarios where there is a need to integrate and automate data flows across diverse systems, such as IoT (Internet of Things) data collection, data migration, log aggregation, and real-time data processing. Its visual interface, extensibility, and scalability make it a popular choice for organizations dealing with complex data integration challenges.

II. NiFi ARCHITECTURE

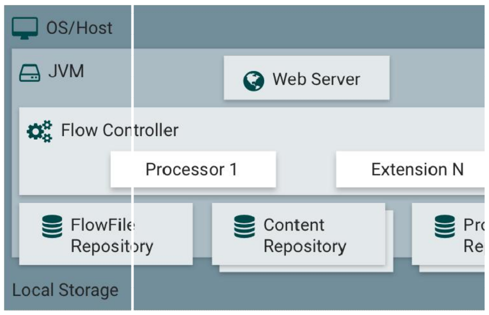

Figure 1:Ni-Fi Architecture

NiFi executes within a JVM on a host operating system. The primary components of NiFi on the JVM are as follows:

1. Web Server

The purpose of the web server is to host NiFi's HTTP-based command and control API.

2. Flow Controller

The flow controller is the brains of the operation. It provides threads for extensions to run on, and manages the schedule of when extensions receive resources to execute.

3. Extensions

There are various types of NiFi extensions which are described in other documents. The key point here is that extensions operate and execute within the JVM.

4. FlowFile Repository

The FlowFile Repository is where NiFi keeps track of the state of what it knows about a given FlowFile that is presently active in the flow. The implementation of the repository is pluggable. The default approach is a persistent Write-Ahead Log located on a specified disk partition.

5. Content Repository

The Content Repository is where the actual content bytes of a given FlowFile live. The implementation of the repository is pluggable. The default approach is a fairly simple mechanism, which stores blocks of data in the file system. More than one file system storage location can be specified so as to get different physical partitions engaged to reduce contention on any single volume.

6. Provenance Repository

The Provenance Repository is where all provenance event data is stored. The repository construct is pluggable with the default implementation being to use one or more physical disk volumes. Within each location event data is indexed and searchable.

NiFi is also able to operate within a cluster.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 3, December 2023

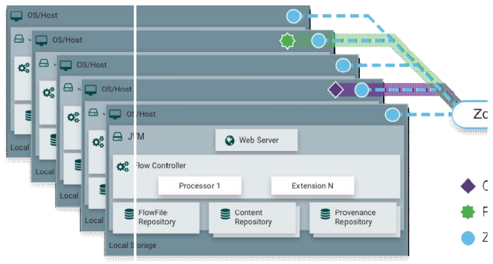

Figure 2: Components of Apache NI-FI

Starting with the NiFi 1.0 release, a Zero-Leader Clustering paradigm is employed. Each node in a NiFi cluster performs the same tasks on the data, but each operates on a different set of data. Apache ZooKeeperelects a single node as the Cluster Coordinator, and failover is handled automatically by ZooKeeper. All cluster nodes report heartbeat and status information to the Cluster Coordinator. The Cluster Coordinator is responsible for disconnecting and connecting nodes. Additionally, every cluster has one Primary Node, also elected by ZooKeeper. As a DataFlow manager, you can interact with the NiFi cluster through the user interface (UI) of any node. Any change you make is replicated to all nodes in the cluster, allowing for multiple entry points.

Performance Expectations and Characteristics of NiFi

NiFi is designed to fully leverage the capabilities of the underlying host system on which it is operating. This maximization of resources is particularly strong with regard to CPU and disk.

For IO

The throughput or latency one can expect to see varies greatly, depending on how the system is configured. Given that there are pluggable approaches to most of the major NiFi subsystems, performance depends on the implementation. But, for something concrete and broadly applicable, consider the out-of-the-box default implementations. These are all persistent with guaranteed delivery and do so using local disk. So being conservative, assume roughly 50 MB per second read/write rate on modest disks or RAID volumes within a typical server. NiFi for a large class of dataflows then should be able to efficiently reach 100 MB per second or more of throughput. That is because linear growth is expected for each physical partition and content repository added to NiFi. This will bottleneck at some point on the FlowFile repository and provenance repository. We plan to provide a benchmarking and performance test template to include in the build, which allows users to easily test their system and to identify where bottlenecks are, and at which point they might become a factor. This template should also make it easy for system administrators to make changes and to verify the impact.

For CPU

The Flow Controller acts as the engine dictating when a particular processor is given a thread to execute. Processors are written to return the thread as soon as they are done executing a task. The Flow Controller can be given a configuration value indicating available threads for the various thread pools it maintains. The ideal number of threads to use depends on the host system resources in terms of numbers of cores, whether that system is running other services as well, and the nature of the processing in the flow. For typical IO-heavy flows, it is reasonable to make many dozens of threads to be available.

For RAM

NiFi lives within the JVM and is thus limited to the memory space it is afforded by the JVM. JVM garbage collection becomes a very important factor to both restricting the total practical heap size, as well as optimizing how well the application runs over time. NiFi jobs can be I/O intensive when reading the same content regularly. Configure a large enough disk to optimize performance.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 3, December 2023

Work flow of apache NiFi
The workflow in Apache NiFi involves designing and managing the flow of data through a series of processors and other components. NiFi provides a graphical user interface that enables users to design data flows using a drag-and-drop approach. Here is a simplified overview of the workflow in Apache NiFi:

Processor Configuration:
Users start by selecting and configuring processors, which represent various actions to be performed on the data. Processors can handle tasks such as data ingestion, transformation, enrichment, routing, and interaction with external systems.

Connection Setup:
After configuring processors, users define connections between them to specify the flow of data. Connections determine how data moves from the output of one processor to the input of another. Users can configure properties of connections, such as backpressure settings.

Input and Output Ports:
Users can set up Input and Output Ports to define entry and exit points for data into and out of the NiFi data flow. Ports provide interfaces for external systems to connect with NiFi.

Data Ingestion:
Processors responsible for data ingestion read data from various sources, such as files, databases, or streaming data. Input data is encapsulated in FlowFiles, which represent individual units of data in NiFi.

Data Transformation and Enrichment:
As FlowFiles move through the data flow, processors can perform transformations on the data, converting formats, enriching content, or applying other data manipulation tasks.

Routing and Conditional Logic:
NiFi supports conditional routing, allowing users to define rules based on content, attributes, or external conditions. This enables dynamic decision-making in the data flow.

Data Movement and Interaction:
Processors can interact with external systems, databases, APIs, or other endpoints to send or receive data. NiFi facilitates the seamless movement of data between different systems.

Data Provenance:
Throughout the data flow, NiFi captures provenance data, which provides a detailed history of how each FlowFile has moved through the system. Provenance data is useful for troubleshooting, auditing, and understanding data lineage.

Monitoring and Management:
NiFi provides a web-based interface for monitoring the status of the data flow, tracking performance metrics, and managing the configuration of processors and connections. Users can monitor the health of the system and make real-time adjustments.

Error Handling and Logging:
NiFi includes features for handling errors, logging information, and notifying users of any issues during data processing. Users can configure processors to manage errors and retries.

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 3, December 2023

Security and Access Control:
NiFi includes security features for user authentication, authorization, and data encryption. Access control mechanisms ensure that only authorized users can interact with and configure the data flow.

Extensibility and Customization:
NiFi is extensible, allowing users to create custom processors, controller services, and reporting tasks to meet specific data integration requirements.

High Level Overview of Key NiFi Features
This sections provides a 20,000 foot view of NiFi's cornerstone fundamentals, so that you can understand the Apache NiFi big picture, and some of its the most interesting features. The key features categories include flow management, ease of use, security, extensible architecture, and flexible scaling model.

Flow Management
A core philosophy of NiFi has been that even at very high scale, guaranteed delivery is a must. This is achieved through effective use of a purpose-built persistent write-ahead log and content repository. Together they are designed in such a way as to allow for very high transaction rates, effective load-spreading, copy-on-write, and play to the strengths of traditional disk read/writes.

Prioritized Queuing
NiFi allows the setting of one or more prioritization schemes for how data is retrieved from a queue. The default is oldest first, but there are times when data should be pulled newest first, largest first, or some other custom scheme.

Ease of Use
Dataflows can become quite complex. Being able to visualize those flows and express them visually can help greatly to reduce that complexity and to identify areas that need to be simplified. NiFi enables not only the visual establishment of dataflows but it does so in real-time. Rather than being 'design and deploy' it is much more like molding clay. If you make a change to the dataflow that change immediately takes effect. Changes are fine-grained and isolated to the affected components. You don't need to stop an entire flow or set of flows just to make some specific modification.

Data Provenance
NiFi automatically records, indexes, and makes available provenance data as objects flow through the system even across fan-in, fan-out, transformations, and more. This information becomes extremely critical in supporting compliance, troubleshooting, optimization, and other scenarios.

Recovery / Recording a rolling buffer of fine-grained history
NiFi's content repository is designed to act as a rolling buffer of history. Data is removed only as it ages off the content repository or as space is needed. This combined with the data provenance capability makes for an incredibly useful basis to enable click-to-content, download of content, and replay, all at a specific point in an object's lifecycle which can even span generations.

Security
A dataflow is only as good as it is secure. NiFi at every point in a dataflow offers secure exchange through the use of protocols with encryption such as 2-way SSL. In addition NiFi enables the flow to encrypt and decrypt content and use shared-keys or other mechanisms on either side of the sender/recipient equation.

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 3, December 2023

III. APPLICATION AND FUTURE SCOPE

However, based on the general trends and the trajectory of data integration technologies, we can make some educated predictions about the future scope of Apache NiFi:

Continued Growth and Adoption:
Apache NiFi has seen steady growth in adoption due to its user-friendly interface, visual data flow design, and scalability. It's likely that the tool will continue to gain popularity as organizations recognize the need for efficient data integration solutions.

Enhancements and New Features:
The Apache NiFi community regularly releases new versions with enhancements, bug fixes, and new features. Future releases may include improvements to the user interface, expanded processor libraries, and additional capabilities to meet evolving data integration challenges.

Integration with Emerging Technologies:
As new technologies and data sources emerge, Apache NiFi is likely to integrate with them. This could include enhanced support for cloud services, integration with emerging data storage and processing technologies, and compatibility with the latest industry standards.

Advanced Data Governance and Security:
With a growing emphasis on data governance and security, future versions of Apache NiFi may include advanced features for data lineage tracking, improved auditing capabilities, and enhanced security measures to meet the compliance requirements of various industries.

AI and Machine Learning Integration:
Integration with artificial intelligence (AI) and machine learning (ML) tools is a trend in the data integration space. Future versions of Apache NiFi may include features or processors that make it easier to incorporate AI and ML models into data flows for real-time processing and decision-making.

Enhanced Real-Time Processing:
As the demand for real-time data processing increases, Apache NiFi may continue to improve its capabilities for handling streaming data. This could include optimizations for low-latency processing, integration with stream processing frameworks, and improved support for event-driven architectures.

IV. CONCLUSION

In conclusion, Apache NiFi is a powerful and versatile open-source data integration tool designed to facilitate the automated movement and processing of data across diverse systems. Its key strengths lie in its user-friendly interface, visual data flow design, and scalability. In summary, Apache NiFi continues to play a significant role in simplifying and automating data integration processes, making it an attractive choice for organizations dealing with complex data movement and transformation requirements. Its ongoing development, community support, and adaptability position it as a valuable tool in the data integration landscape.

REFERENCES

[1]. Apache NiFi

[2]. The official website is the central hub for Apache NiFi information, including downloads, documentation, news, and community resources.

[3]. Apache NiFi Documentation

[4]. The official documentation provides comprehensive guides, tutorials, and reference material covering installation, configuration, and usage of Apache NiFi.

[5]. Apache NiFi GitHub Repository
[6]. The GitHub repository hosts the source code, release notes, and issues related to Apache NiFi. It's a valuable resource for developers and those interested in the project's development.
[7]. Apache NiFi Users Mailing List
[8]. The mailing list is a forum for NiFi users to ask questions, share experiences, and seek help from the community. It's a good place to connect with other NiFi users and developers.
[9]. NiFi Registry Documentation
[10]. If you are using NiFi Registry as part of your workflow, its documentation provides information on version control, managing flows, and deploying data flows.
[11]. Apache NiFi Wiki
[12]. The NiFi Wiki on the Apache Confluence site includes additional resources, tips, and community-contributed content.
[13]. Apache NiFi Questions on Stack Overflow
[14]. Stack Overflow is a popular platform for asking and answering technical questions. The Apache NiFi tag is used for discussions related to NiFi on this platform.
[15]. Apache NiFi Blogs
[16]. The Apache NiFi Blogs page features blog posts related to Apache NiFi. It can be a source of additional insights, use cases, and community contributions.
[17]. NiFi Processor Documentation
[18]. This example link directs you to the documentation for the ExecuteStreamCommand processor. You can explore other processor documentation from the main documentation page.
[19]. [iot] Wikipedia. Internet of Things [online]. Retrieved: from: http://en.wikipedia.org/wiki/Internet_of_Things
[20]. bigdata] Wikipedia. Big Data [online]. Retrieved:, from: http://en.wikipedia.org/wiki/Big_data