Six Questions Your IoT Solution Must Answer & How Apache MiNiFi and NiFi Help

Andrew Psaltis - @itmdata
EVERY BUSINESS IS A DATA BUSINESS

MASTER THE VALUE OF DATA

EMBRACE AN OPEN APPROACH
Industrial processes create large amounts of data

Always on, always connected devices generate a constant stream of data related to the operations of industrial businesses.

These datasets contain:
- What happened?
- Why something happened or not?
- Quantification of events

These datasets go by many names:
- “SCADA Data”
- “Control System Data”
- “Historian Data”
- “Machine Data”
- “Measurement Logs”

How are my …
- People?
- Processes?
- Equipment?
The challenge:

Collect 20k time series sampled at 1Hz from the control systems of critical pieces of equipment.

Deliver this data in its raw format to a platform that can service both data scientist and BI analysis.

Oh yeah, and there is this satellite thing ...
IoT World Forum Reference Model

Levels

7. Collaboration & Processes
   (Involving People & Business Processes)

6. Application
   (Reporting, Analytics, Control)

5. Data Abstraction
   (Aggregation & Access)

4. Data Accumulation
   (Storage)

3. Edge Computing
   (Data Element Analysis & Transformation)

2. Connectivity
   (Communication & Processing Units)

1. Physical Devices & Controllers
   (The “Things” in IoT)
IoT Reference Model Objectives

Levels

7
6
5
4
3
2
1

Center

IT
Query Based
Data at Rest
Non-real Time

OT
Event Based
Data in Motion
Real Time

Edge
Sensors, Devices, Machines, Intelligent Edge Nodes of all types

© 2015 Cloudera and/or its affiliates. All rights reserved.
OT / IT Convergence – Must Occur to Achieve Business Improvement

**Operational technology system**
- Managing assets, controlling technology processes
- Simple data type, just-in-time analytics, high data rate

- Control-room applications (industrial control configuration software)
- Factory manager
- Real-time database
- Over 1 million messages/sec

**Information technology system**
- Managing information, automating business processes
- Complicated data type, deep analytics, low data rate

- Enterprise resource planning
- Customer relationship management
- Business intelligence
- Enterprise service bus

**Physical infrastructure**
- Sensors and actuators
- Field engineer
- SCADA

**Figure 2**: To achieve a wider range of business-improvement opportunities through analysis and optimization, field OT and corporate IT systems must be integrated.

Minimum Requirements for IoT Solution

- Security
- Provenance
- Connectivity
- Prioritization
- Extensibility
- Real-Time
Apache NiFi
Simplistic View of DataFlows: Easy, Definitive
Realistic View of Dataflow: Complex, Convoluted
Moving data **effectively** is hard

Standards: http://xkcd.com/927/
Apache NiFi

Dataflow

- Web-based User Interface for creating, monitoring, & controlling data flows
- Directed graphs of data routing and transformation
- Highly configurable - modify data flow at runtime, dynamically prioritize data
- Easily extensible through development of custom components
- Data Provenance tracks data through entire system

NiFi - Terminology

- **FlowFile**
  - Unit of data moving through the system
  - Content + Attributes (key/value pairs)
# FlowFiles are like HTTP data

<table>
<thead>
<tr>
<th>HTTP Data</th>
<th>FlowFile</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP/1.1 200 OK</td>
<td>Standard FlowFile Attributes</td>
</tr>
<tr>
<td>Date: Sun, 10 Oct 2010 23:26:07 GMT</td>
<td>Key: 'entryDate' Value: ‘Fri Jun 17 17:15:04 EDT 2016’</td>
</tr>
<tr>
<td>Server: Apache/2.2.8 (CentOS) OpenSSL/0.9.8g</td>
<td>Key: ‘lineageStartDate’ Value: ‘Fri Jun 17 17:15:04 EDT 2016’</td>
</tr>
<tr>
<td>Last-Modified: Sun, 26 Sep 2010 22:04:35 GMT</td>
<td>Key: ‘fileSize’ Value: ’23609’</td>
</tr>
<tr>
<td>ETag: ”45b6-834-49130cc1182c0”</td>
<td>FlowFile Attribute Map Content</td>
</tr>
<tr>
<td>Accept-Ranges: bytes</td>
<td>Key: ‘filename’ Value: ’15650246997242’</td>
</tr>
<tr>
<td>Connection: close</td>
<td>Key: ‘path’ Value: ‘./’</td>
</tr>
<tr>
<td>Content-Length: 13</td>
<td></td>
</tr>
<tr>
<td>Content-Type: text/html</td>
<td>Binary Content *</td>
</tr>
<tr>
<td>Hello world!</td>
<td></td>
</tr>
</tbody>
</table>

## Header

## Content
FlowFiles & Data Agnosticism

- NiFi is data agnostic!
- But, NiFi was designed understanding that users can care about specifics and provides tooling to interact with specific formats, protocols, etc.

Robustness principle

Be conservative in what you do, be liberal in what you accept from others

NiFi - Terminology

- **FlowFile**
  - Unit of data moving through the system
  - Content + Attributes (key/value pairs)

- **Processor**
  - Performs the work, can access FlowFiles

- **Connection**
  - Links between processors
  - Queues that can be dynamically prioritized

- **Process Group**
  - Set of processors and their connections
  - Receive data via input ports, send data via output ports
NiFi is based on Flow Based Programming (FBP)

<table>
<thead>
<tr>
<th>FBP Term</th>
<th>NiFi Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Packet</td>
<td>FlowFile</td>
<td>Each object moving through the system.</td>
</tr>
<tr>
<td>Black Box</td>
<td>FlowFile Processor</td>
<td>Performs the work, doing some combination of data routing, transformation, or mediation between systems.</td>
</tr>
<tr>
<td>Bounded Buffer</td>
<td>Connection</td>
<td>The linkage between processors, acting as queues and allowing various processes to interact at differing rates.</td>
</tr>
<tr>
<td>Scheduler</td>
<td>Flow Controller</td>
<td>Maintains the knowledge of how processes are connected, and manages the threads and allocations thereof which all processes use.</td>
</tr>
<tr>
<td>Subnet</td>
<td>Process Group</td>
<td>A set of processes and their connections, which can receive and send data via ports. A process group allows creation of entirely new component simply by composition of its components.</td>
</tr>
</tbody>
</table>
The need for data provenance

For Operators
• Traceability, lineage
• Recovery and replay

For Compliance
• Audit trail
• Remediation

For Business / Mission
• Value sources
• Value IT investment
### Data Provenance (Not just Lineage)

- View attributes and content at given points in time (before and after each processor) !!!
- Records, indexes, and makes events available for display

#### NiFi Flow Data Provenance

**Oldest event available: 07/29/2015 14:08:06 EDT**

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Type</th>
<th>FlowFile Uuid</th>
<th>Size</th>
<th>Component Name</th>
<th>Component Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>07/29/2015 16:21:34.368 EDT</td>
<td>DROP</td>
<td>3b9f20bc-031e-4af8-ad8a-fedce...</td>
<td>158 bytes</td>
<td>PutSolrContentStream</td>
<td>PutSolrContentStream</td>
</tr>
<tr>
<td>07/29/2015 16:21:34.367 EDT</td>
<td>SEND</td>
<td>3b9f20bc-031e-4af8-ad8a-fedce...</td>
<td>158 bytes</td>
<td>PutSolrContentStream</td>
<td>PutSolrContentStream</td>
</tr>
<tr>
<td>07/29/2015 16:21:34.366 EDT</td>
<td>DROP</td>
<td>6f5036bc-1768-476d-960d-1f8d...</td>
<td>2.15 KB</td>
<td>PutSolrContentStream</td>
<td>PutSolrContentStream</td>
</tr>
</tbody>
</table>

#### Provenance Event

<table>
<thead>
<tr>
<th>Details</th>
<th>Attributes</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>07/29/2015 16:21:34.367 EDT</td>
<td></td>
</tr>
<tr>
<td>Event Duration</td>
<td>00:00:00.001</td>
<td></td>
</tr>
<tr>
<td>Lineage Duration</td>
<td>00:00:00.117</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>SEND</td>
<td></td>
</tr>
<tr>
<td>FlowFile Uuid</td>
<td>3b9f20bc-031e-4af8-ad8a-fedce4e0999</td>
<td></td>
</tr>
<tr>
<td>File Size</td>
<td>158 bytes</td>
<td></td>
</tr>
<tr>
<td>Component Id</td>
<td>fa7b551f-405-4fde-b094-0b0d69c03472</td>
<td></td>
</tr>
<tr>
<td>Component Name</td>
<td>PutSolrContentStream</td>
<td></td>
</tr>
<tr>
<td>Component Type</td>
<td>PutSolrContentStream</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>No value set</td>
<td></td>
</tr>
</tbody>
</table>
Provenance

Types of Lineage
- Event (runtime)
- Configuration (design time)
The need for fine-grained security and compliance

It’s not enough to say you have encrypted communications

- Enterprise authorization services – entitlements change often
- People and systems with different roles require different access levels
- Tagged/classified data
Security

**SOURCES**
- Sign, encrypt, static (data and control)

**REGIONAL INFRASTRUCTURE**
- TLS, obfuscation, dynamic entitlements
- Kerberos, PKI, AD/DS, etc.

**CORE INFRASTRUCTURE**
- Hybrid – cloud/on-premises
- Low-latency
- Global context

- Constrained
- High-latency
- Localized context
Data Buffering w/ Back Pressure and Pressure Release

- Supports buffering of all queued data.
- Ability to back-pressure (Even if there is no load balancing, nodes can say “Back-Off” and other nodes in the pipeline pick up the slack.
- When backpressure is applied to a connection, it will cause the processor that is the source of the connection to stop being scheduled to run until the queue clears out. However, data will still queue up in that processor's incoming connections.
Data Buffering w/ Back Pressure and Pressure Release

- GenerateFlowFile
  - In: 0 (0 bytes), 5 min
  - Read/Write: 0 bytes / 0 bytes, 5 min
  - Out: 0 (0 bytes), 5 min
  - Tasks/Time: 0 / 00:00:00.000, 5 min

- RouteOnAttribute
  - In: 0 (0 bytes), 5 min
  - Read/Write: 0 bytes / 0 bytes, 5 min
  - Out: 0 (0 bytes), 5 min
  - Tasks/Time: 0 / 00:00:00.000, 5 min

Name success
Queued: 10,000 (0 bytes)
Guaranteed Data Delivery

• Even at very high scale, delivery is guaranteed
• Persistent Write Ahead Log (Flow File Repository) and Data Partitioning (Content Repository) ensures this. They are together designed in a way that they allow:
  • Very high transaction rates
  • Effective load spreading
  • Copy-on-write scheme (for every change in data)
  • Pass-by-reference
Prioritized Queuing

- NiFi allows the setting of one or more prioritization schemes for how data is retrieved from a queue.
- Oldest First, Newest first, Largest first, Smallest First, or custom scheme
- The default is oldest first
NiFi - Queue Prioritization

• Configure a prioritizer per connection

• Determine what is important for your data – time based, arrival order, importance of a data set

• Funnel many connections down to a single connection to prioritize across data sets

• Develop your own prioritizer if needed
Designed for Extension

✓ Processor - Push/Pull behavior. Custom UI
✓ Controller Service - Used to enable reusable components / shared services throughout the flow
✓ Reporting Tasks - Used to push data from NiFi to some external service (metrics, provenance, etc..)
✓ Prioritizer
✓ User Interface
✓ REST API

• These extensions are bundles in something called as NAR Files (NiFi Archives).
NiFi Positioning

- **Enterprise Service Bus**
  (Fuse, Mule, etc.)

- **Processing Framework**
  (Storm, Spark, etc.)

- **ETL**
  (Informatica, etc.)

- **Messaging Bus**
  (Kafka, MQ, etc.)

**Apache NiFi / MiNiFi**
Apache NiFi / Processing Frameworks

NiFi

Simple event processing
- Primarily feed data into processing frameworks, can process data, with a focus on simple event processing
- Operate on a single piece of data, or in correlation with an enrichment dataset (enrichment, parsing, splitting, and transformations)
- Can scale out, but scale up better to take full advantage of hardware resources, run concurrent processing tasks/threads (processing terabytes of data per day on a single node)

⚠ Not another distributed processing framework, but to feed data into those

Processing Frameworks (Storm, Spark, etc.)

Complex and distributed processing
- Complex processing from multiple streams (JOIN operations)
- Analyzing data across time windows (rolling window aggregation, standard deviation, etc.)
- Scale out to thousands of nodes if needed

⚠ Not designed to collect data or manage data flow
Apache NiFi / Messaging Bus Services

NiFi

Provide dataflow solution

- Centralized management, from edge to core
- Great traceability, event level data provenance starting when data is born
- Interactive command and control – real time operational visibility
- Dataflow management, including prioritization, back pressure, and edge intelligence
- Visual representation of global dataflow

⚠ Not a messaging bus, flow maintenance needed when you have frequent consumer side updates

Messaging Bus (Kafka, JMS, etc.)

Provide messaging bus service

- Low latency
- Great data durability
- Decentralized management (producers & consumers)
- Low broker maintenance for dynamic consumer side updates

⚠ Not designed to solve dataflow problems (prioritization, edge intelligence, etc.)

⚠ Traceability limited to in/out of topics, no lineage

⚠ Lack of global view of components/connectivities
Apache NiFi / Integration, or ingestion, Frameworks

NiFi

End user facing dataflow management tool

- Out of the box solution for dataflow management
- Interactive command and control in the core, design and deploy on the edge
- Flexible failure handling at each point of the flow
- Visual representation of global dataflow and connectivities
- Native cross data center communication
- Data provenance for traceability

⚠ Not a library to be embedded in other applications

Integration framework (Spring Integration, Camel, etc), ingestion framework (Flume, etc)

Developer facing integration tool with a focus on data ingestion

- A set of tools to orchestrate workflow
- A fixed design and deploy pattern
- Leverage messaging bus across disconnected networks

⚠ Developer facing, custom coding needed to optimize
⚠ Pre-built failure handling, lack of flexibility
⚠ No holistic view of global dataflow
⚠ No built-in data traceability
Apache NiFi / ETL Tools

NiFi

NOT schema dependent

• Dataflow management for both structured and unstructured data, powered by separation of metadata and payload
• Schema is not required, but you can have schema
• Minimum modeling effort, just enough to manage dataflows
• Do the plumbing job, maximize developers’ brainpower for creative work

⚠ Not designed to do heavy lifting transformation work for DB tables (JOIN datasets, etc.). You can create custom processors to do that, but long way to go to catch up with existing ETL tools from user experience perspective (GUI for data wrangling, cleansing, etc.)

ETL (Informatica, etc.)

Schema dependent

• Tailored for Databases/WH
• ETL operations based on schema/data modeling
• Highly efficient, optimized performance

⚠ Must pre-prepare your data, time consuming to build data modeling, and maintain schemas
⚠ Not geared towards handling unstructured data, PDF, Audio, Video, etc.
⚠ Not designed to solve dataflow problems
Key Features and Principles

- Guaranteed delivery
- Data buffering
  - Backpressure
  - Pressure release
- Prioritized queuing
- Flow specific QoS
  - Latency vs. throughput
  - Loss tolerance
- Data provenance
- Recovery/recording a rolling log of fine-grained history
- Visual command and control
- Flow templates
- Pluggable/multi-role security
- Designed for extension
- Clustering
Apache MiNiFi
Apache MiNiFi

“Let me get the key parts of NiFi close to where data begins and provide bidirectional data transfer"

- NiFi lives in the data center. Give it an enterprise server or a cluster of them.
- MiNiFi lives as close to where data is born and is a guest on that device or system
Apache NiFi MiNiFi

Key Features

• Guaranteed delivery
• Data buffering
  – Backpressure
  – Pressure release
• Prioritized queuing
• Flow specific QoS
  – Latency vs. throughput
  – Loss tolerance
• Data provenance

• Recovery/recording a rolling log of fine-grained history
• Designed for extension

• Design and Deploy
• Warm re-deploys
Apache MiNiFi

Realities of computing outside the comforts of the data center

- Limited computing capability
- Limited power/network
- Restricted software library/platform availability
- No UI
- Physically inaccessible
- Not frequently updated
- Competing standards/protocols
- Scalability
- Privacy & Security
MiNiFi: Precedent from NiFi

A quick look at NiFi Site to Site

- Provides the semantics between two NiFi components across network boundaries
  - A custom protocol for inter-NiFi communication
  - Secure, Extensible, Load Balanced & Scalable Delivery to Cluster

- Extracted out to a client library which powers integration into popular frameworks like Apache Spark, Apache Storm, Apache Flink, and Apache Apex

- Attributes and the FlowFile format maintained

https://nifi.apache.org/docs/nifi-docs/html/user-guide.html#site-to-site
MiNiFi: Precedent from NiFi

A deeper dive into provenance

- Fine-grained, event level access of interactions with FlowFiles
  - CREATE, RECEIVE, FETCH, SEND, DOWNLOAD, DROP, EXPIRE, FORK, JOIN ...

- Captures the associated attributes/metadata at the time of the event

- A map of a FlowFile’s journey and how they relate to other FlowFiles in a system
  - MiNiFi enables us to get more and further illuminate the map of data processing

http://nifi.apache.org/docs/nifi-docs/html/user-guide.html#data-provenance
MiNiFi: Precedent from NiFi

**RECEIVE event**

Provenance Event

<table>
<thead>
<tr>
<th>DETAILS</th>
<th>ATTRIBUTES</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>04/04/2017 16:59:49.642 CEST</td>
<td>Parent FlowFiles (0)</td>
</tr>
<tr>
<td>Event Duration</td>
<td>&lt; 1ms</td>
<td>No parents</td>
</tr>
<tr>
<td>Lineage Duration</td>
<td>&lt; 1ms</td>
<td>Child FlowFiles (0)</td>
</tr>
<tr>
<td>Type</td>
<td>RECEIVE</td>
<td></td>
</tr>
</tbody>
</table>

FlowFile Uuid
e5a2810e-e332-484f-9fd5-df4eb36015f0

File Size
11 bytes

Component Id
397820ce-015b-1000-b89d-7e7d65b8b671

Component Name
ListenHTTP

Component Type
ListenHTTP
Apache MiNiFi

Departures from NiFi in getting the right fit

- The feedback loop is longer and not guaranteed
  - Removal of Web Server and UI

- Declarative configuration
  - Lends itself well to CM processes
  - Extensible interface to support varying formats
    - Currently provided in YAML

- Reduced set of bundled components
Apache MiNiFi: Scoping

Provide all the key principles of NiFi in varying, smaller footprints

- **Go small**: Java – *Write once, run anywhere*  
  - Feature parity and reuse of core NiFi libraries

- **Go smaller**: C++ – *Write once**, run anywhere

- **Go smallest**: *Write n-many times, run anywhere*  
  Language libraries to support tagging, FlowFile format, Site to Site protocol, and provenance generation without a processing framework  
  - Mobile: Android & iOS  
  - Language SDKs
WHAT IS THIS!?

A NiFi FOR ANTS!?!
Collect | Process | Store/Query

Event Time | Access Time
Harnessing Data in Motion

- **Sources**
- **Regional Infrastructure**
- **Core Infrastructure**

- Constrained
- High-latency
- Localized context

- Hybrid – cloud / on-premises
- Low-latency
- Global context
Apache NiFi and MiNiFi Use Case
Apache MiNiFi Use Case: Distributed Systems: Connected Car

1. **MiNiFi**
   - Buffer offline delay-tolerant data
   - Real-time data

2. **DSRC RSU**

3. **Data Storage**

4. **Wi-Fi AP**

5. **Wi-Fi AP**
   - Possibly modify route or adjust sampling rate and prepare to flush delay-tolerant buffers when SDRC RSU or Wi-Fi AP in range

Car Wireless Gateway

Raw data via CAN, Ethernet

1. **SD Card**

2. **LTE**

3. **Map Database**

4. **eNodeB**

5. **Decision-Making Server**
MiNiFi: Use Case - Connected Car

- Outside vehicle’s network firewall
- On telematics layer
The GENIVI Alliance is a nonprofit industry alliance committed to driving the broad adoption of specified, open source, In-Vehicle Infotainment software.
Apache MiNiFi Use Case: Distributed Systems: Retail POS

100’s or 1000’s or more terminals to gather:

- Real-time pricing info
- Real-time inventory updates
- Real-time offers
- Traceable logs of transactions
- Log of user-interactions for trouble-shooting devices
- Predictive help desk
Apache MiNiFi Use Cases: Distributed Systems of Devices

- Routers
- Security cameras
- Cable modems
- ATM
- Fleet of Trucks
- Manufacturing Line
- Security Appliance
- Point of Sale
- Weather detection system
- Thermostats
- Utility/Power meters
- Fleet of Ships

Any distributed system of devices with data to be collected
Apache NiFi and MiNiFi Ecosystem
Apache NiFi: The Ecosystem

Provided a framework to extend the reach of data ingest

- Site-to-Site in MiNiFi instances provides machine-to-machine (M2M) communication
  - Data arrives to NiFi in a transparent manner allowing integration to existing flows

- Similar attention to extensibility in both Java and C++ clients allows agents to fit the needs of your organization

- Reduced footprint allows NiFi functionality to aid in production of high fidelity data, more closely attributable and tracked from where it is generated
Apache NiFi: The Ecosystem

But more instances complicate my operational management!

- Enter the MiNiFi Command & Control
  - Provide tooling to map the UX of interactive command and control in NiFi to the design and deploy approach of MiNiFi

https://cwiki.apache.org/confluence/display/MINIFI/MiNiFi+Command+and+Control
Apache NiFi: The Ecosystem

Building on efforts for reusable components in the community

- Configuration Management of Flows & Versioning
  - The evolution of templates to better support SDLC functions

- Extension Repositories
  - Publish & Share extension bundles (NARs)
  - [https://cwiki.apache.org/confluence/display/NIFI/Extension+Repositories+%28aka+Extension+Registry%29+for+Dynamically-loaded+Extensions](https://cwiki.apache.org/confluence/display/NIFI/Extension+Repositories+%28aka+Extension+Registry%29+for+Dynamically-loaded+Extensions)

- Variable Registry
  - Initial framework support & file-based implementation
  - [https://cwiki.apache.org/confluence/display/NIFI/Variable+Registry](https://cwiki.apache.org/confluence/display/NIFI/Variable+Registry)
Why Apache NiFi & MiNiFi?

- Moving data is multifaceted in its challenges and these are present in different contexts at varying scopes

- Provide common tooling and extensions that are commonly needed but be flexible for extension
  - Leverage existing libraries and expansive Java ecosystem for functionality
  - Allow organizations to integrate with their existing infrastructure

- Empower folks managing your infrastructure to make changes and reason about issues that are occurring
  - Data Provenance to show context and data’s journey
  - User Interface/Experience a key component
• Level 7 – Collaborative & Processes
• Level 6 – Application
• Level 5 – Data Abstraction (Hadoop)
• Level 4 – Data Accumulation
  • Provenance
• Level 3 – Edge Computing
  • Prioritization
  • Command and Control Interface
  • Simple Event Processing
• Level 2 – Connectivity
  • Data Source Agnostic
  • Secure and Guaranteed Bidirectional Delivery
• Level 1 – Edge
  • MiNiFi can be embedded in devices.
Learn more and join us!

Apache NiFi site
https://nifi.apache.org

Subproject MiNiFi site
https://nifi.apache.org/minifi/

Subscribe to and collaborate at
dev@nifi.apache.org
users@nifi.apache.org

Submit Ideas or Issues
https://issues.apache.org/jira/browse/NIFI
https://issues.apache.org/jira/browse/MINIFI

Follow on Twitter
@apachenifi
Thank You