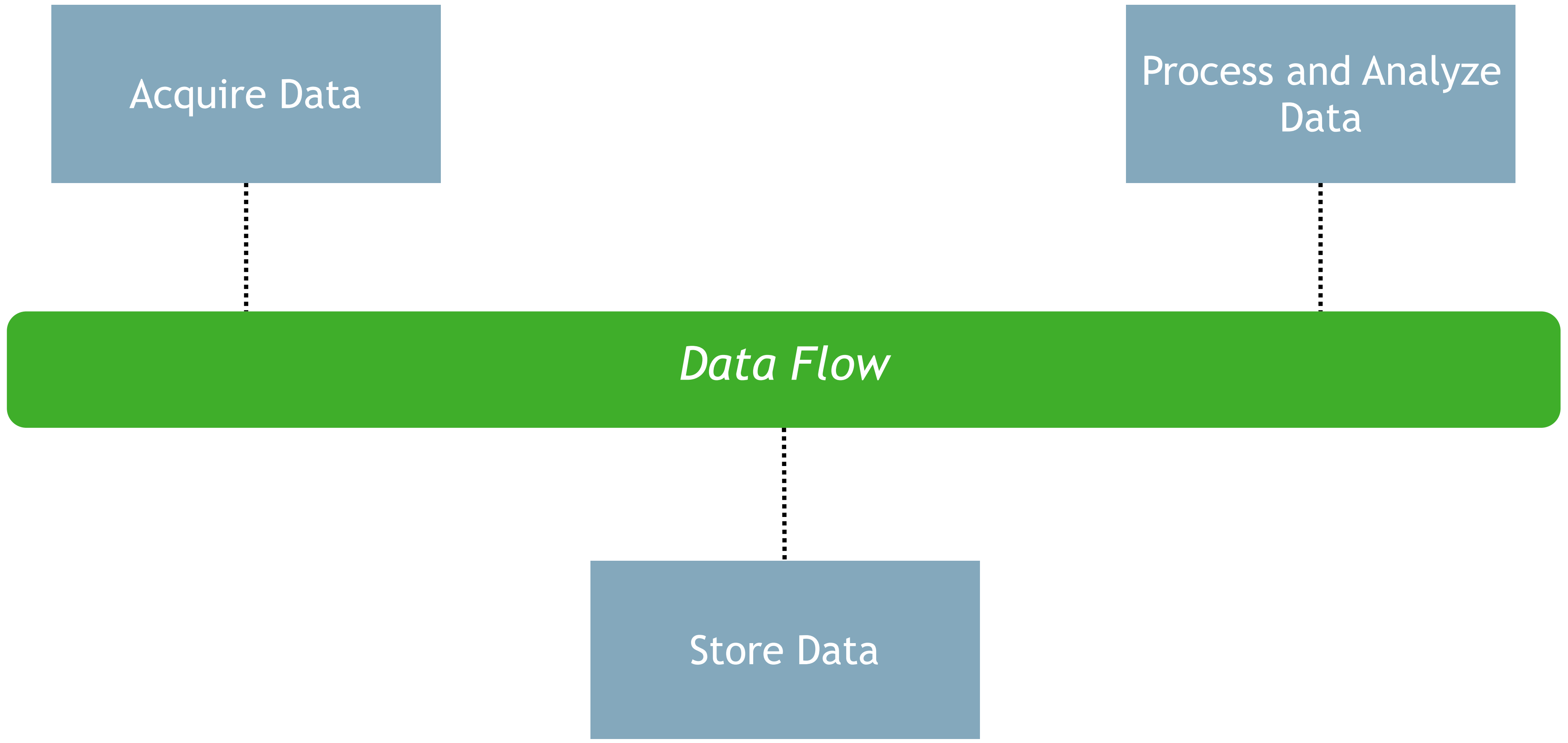
Apache NiFi

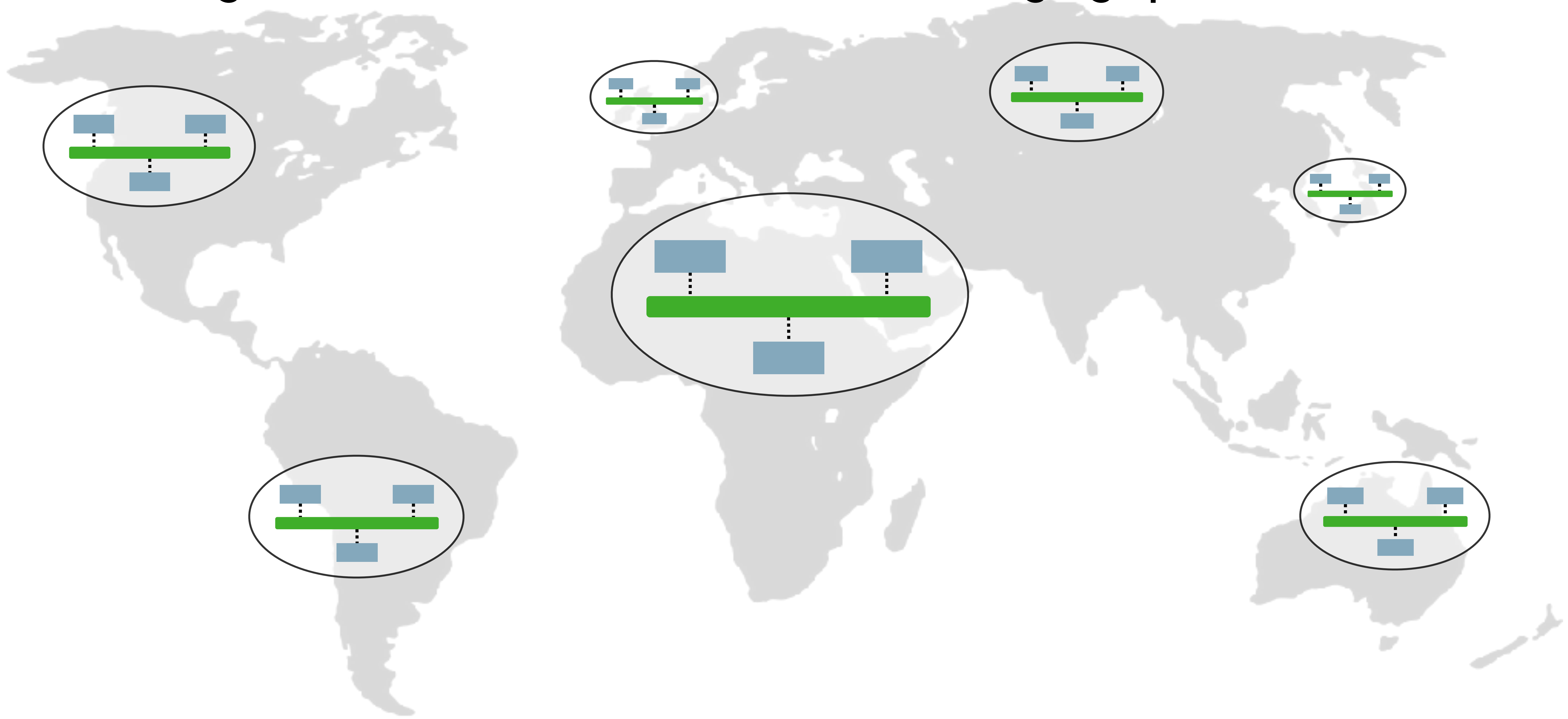
What is Apache NiFi?

Simplistic View of Enterprise Data Flow



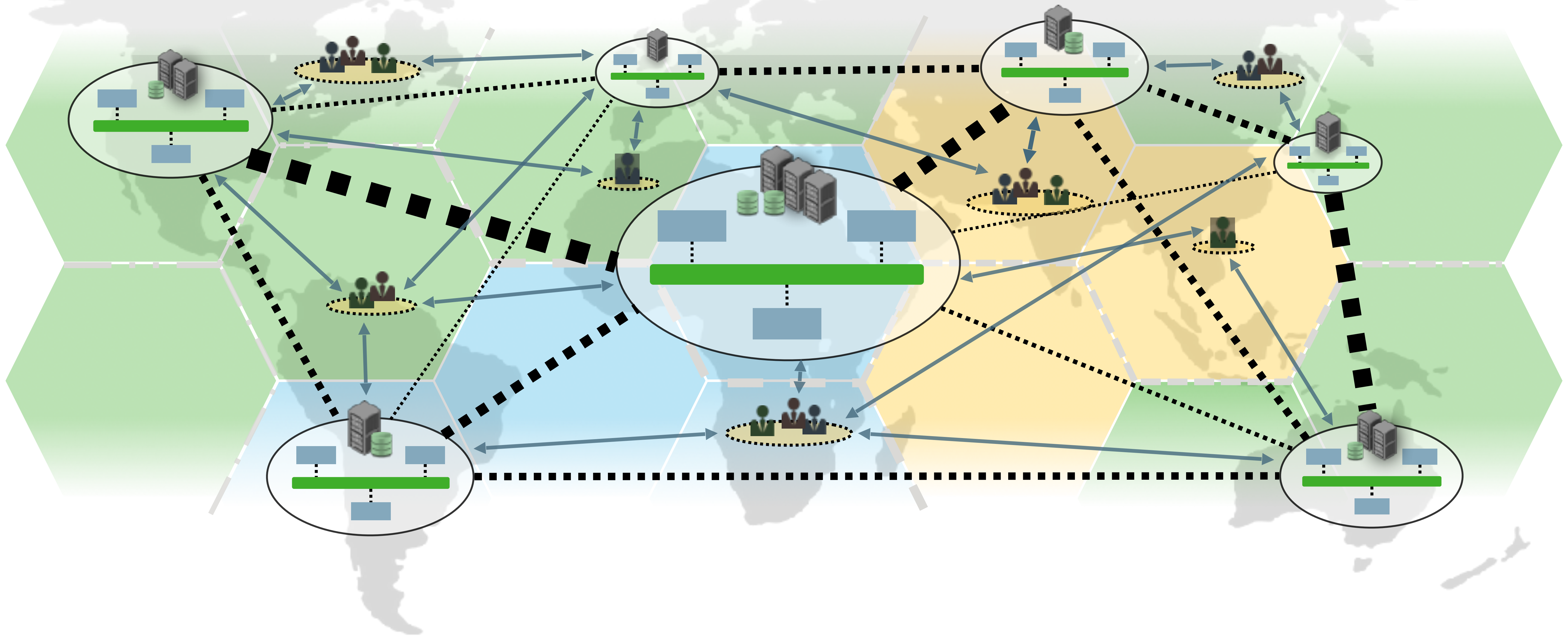
Realistic View of Enterprise Data Flow

Different organizations/business units across different geographic locations…



Realistic View of Enterprise Data Flow

Interacting with different business partners and customers



Apache NiFi

- Created to address the challenges of global enterprise dataflow
- Key features:
  - Visual Command and Control
  - Data Lineage (Provenance)
  - Data Prioritization
  - Data Buffering/Back-Pressure
  - Control Latency vs. Throughput
  - Secure Control Plane / Data Plane
  - Scale Out Clustering
  - Extensibility

Apache NiFi

What is Apache NiFi used for?

- Reliable and secure transfer of data between systems
- Delivery of data from sources to analytic platforms
- Enrichment and preparation of data:
  - Conversion between formats
  - Extraction/Parsing
  - Routing decisions

What is Apache NiFi NOT used for?

- Distributed Computation
- Complex Event Processing
- Joins / Complex Rolling Window Operations

Hadoop Ecosystem Integrations

HDFS Ingest

MergeContent

- Merges into appropriately sized files for HDFS
- Based on size, number of messages, and time

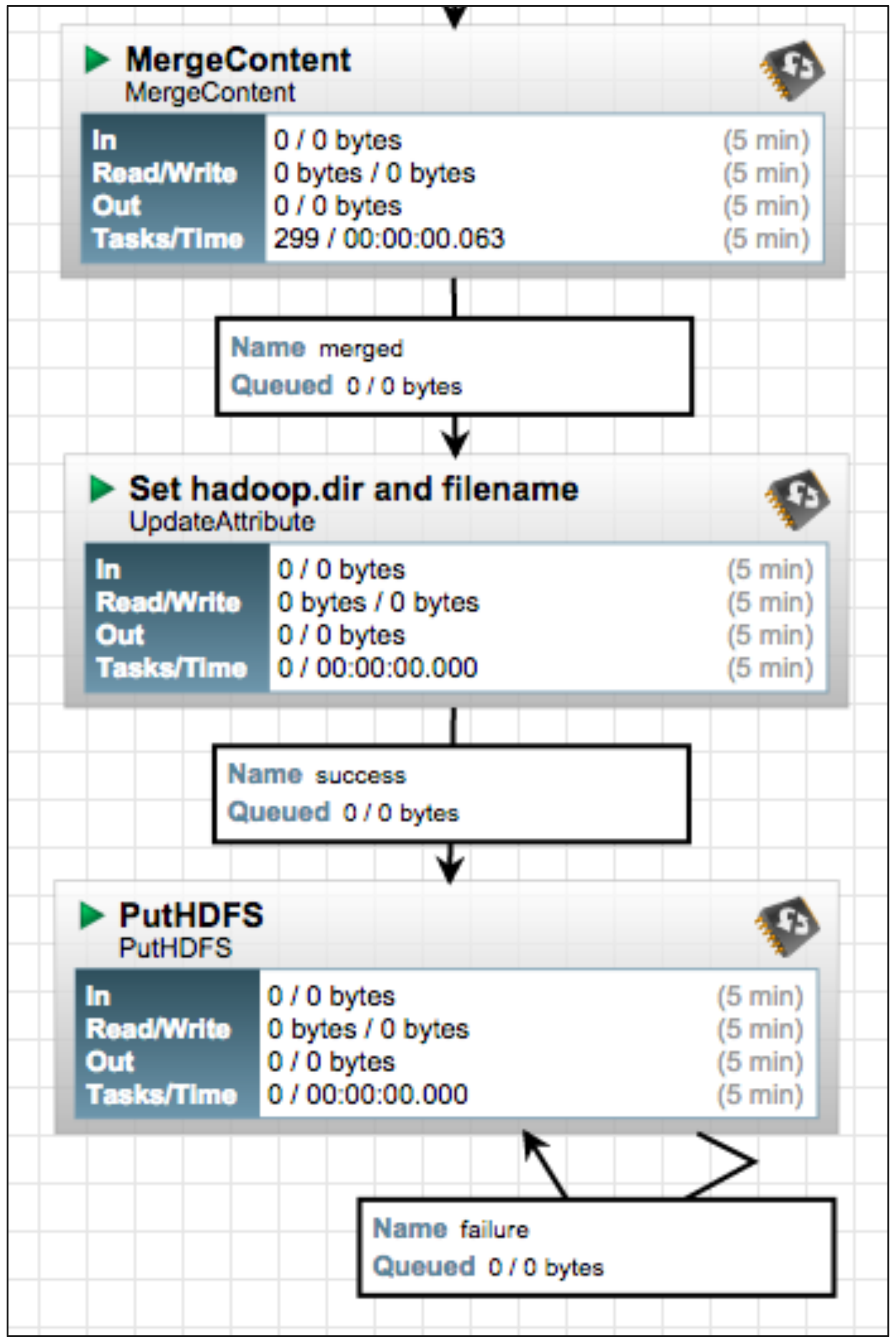
UpdateAttribute

- Sets the HDFS directory and filename
- Use expression language to dynamically bin by date:

```
/data/${now():format('yyyy/MM/dd/HH')}/
```

PutHDFS

- Writes FlowFile content to HDFS
- Supports Conflict Resolution Strategy and Kerberos authentication



MergeContent
MergeContent

In	0 / 0 bytes	(5 min)
Read/Write	0 bytes / 0 bytes	(5 min)
Out	0 / 0 bytes	(5 min)
Tasks/Time	299 / 00:00:00.063	(5 min)

Name merged
Queued 0 / 0 bytes

Set hadoop.dir and filename
UpdateAttribute

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 0 / 0 bytes

PutHDFS
PutHDFS

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 failure
Queued 0 / 0 bytes

HDFS Retrieval

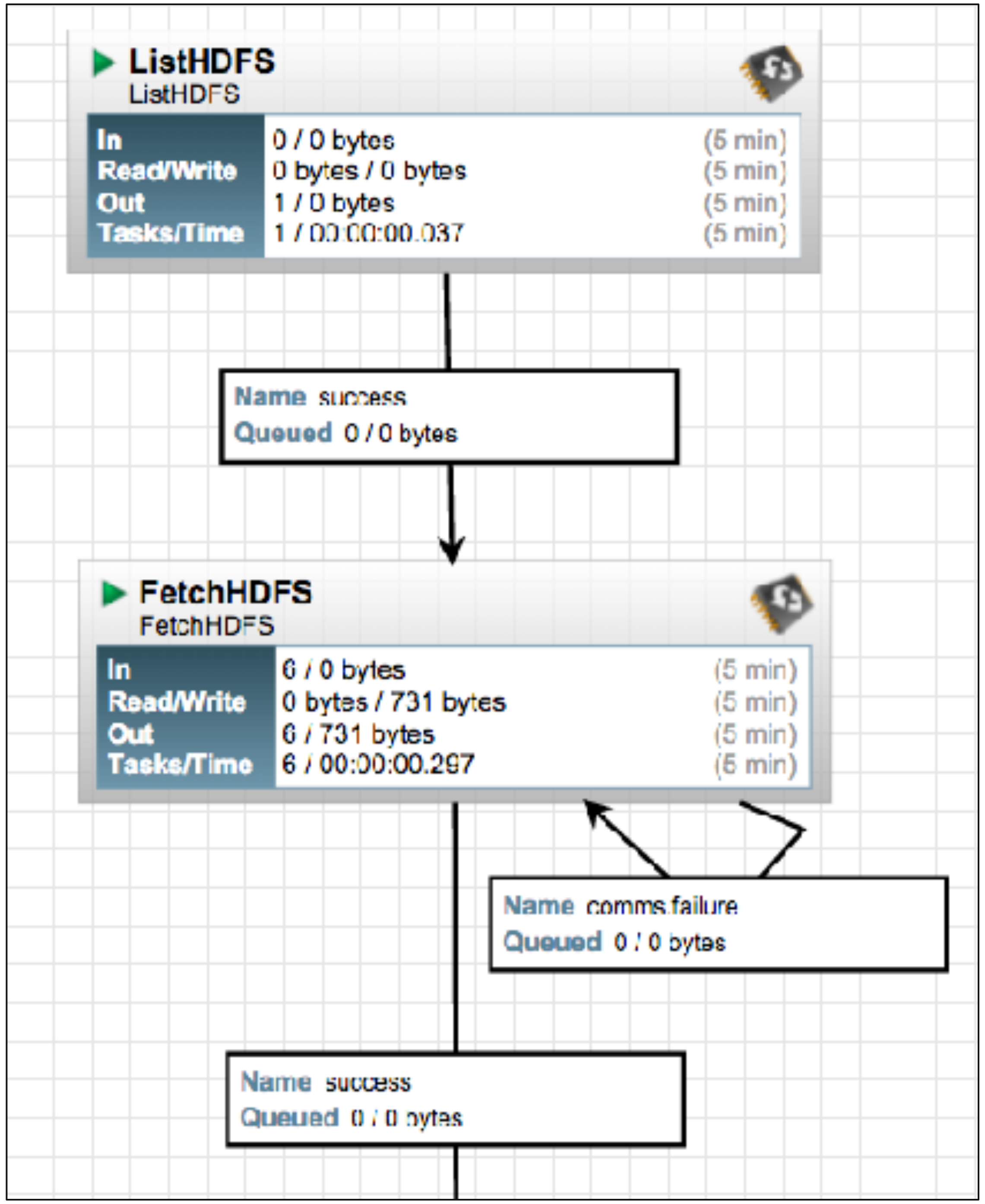
ListHDFS

- Periodically perform listing on HDFS directory
- Produces FlowFile per HDFS file
- Flow only contains HDFS path & filename

FetchHDFS

- Retrieves a file from HDFS
- Use incoming FlowFiles to dynamically fetch:

  HDFS Filename: `${path}/${filename}`



ListHDFS
ListHDFS

In	0 / 0 bytes	(5 min)
Read/Write	0 bytes / 0 bytes	(5 min)
Out	1 / 0 bytes	(5 min)
Tasks/Time	1 / 00:00:00.037	(5 min)

Name success
Queued 0 / 0 bytes

FetchHDFS
FetchHDFS

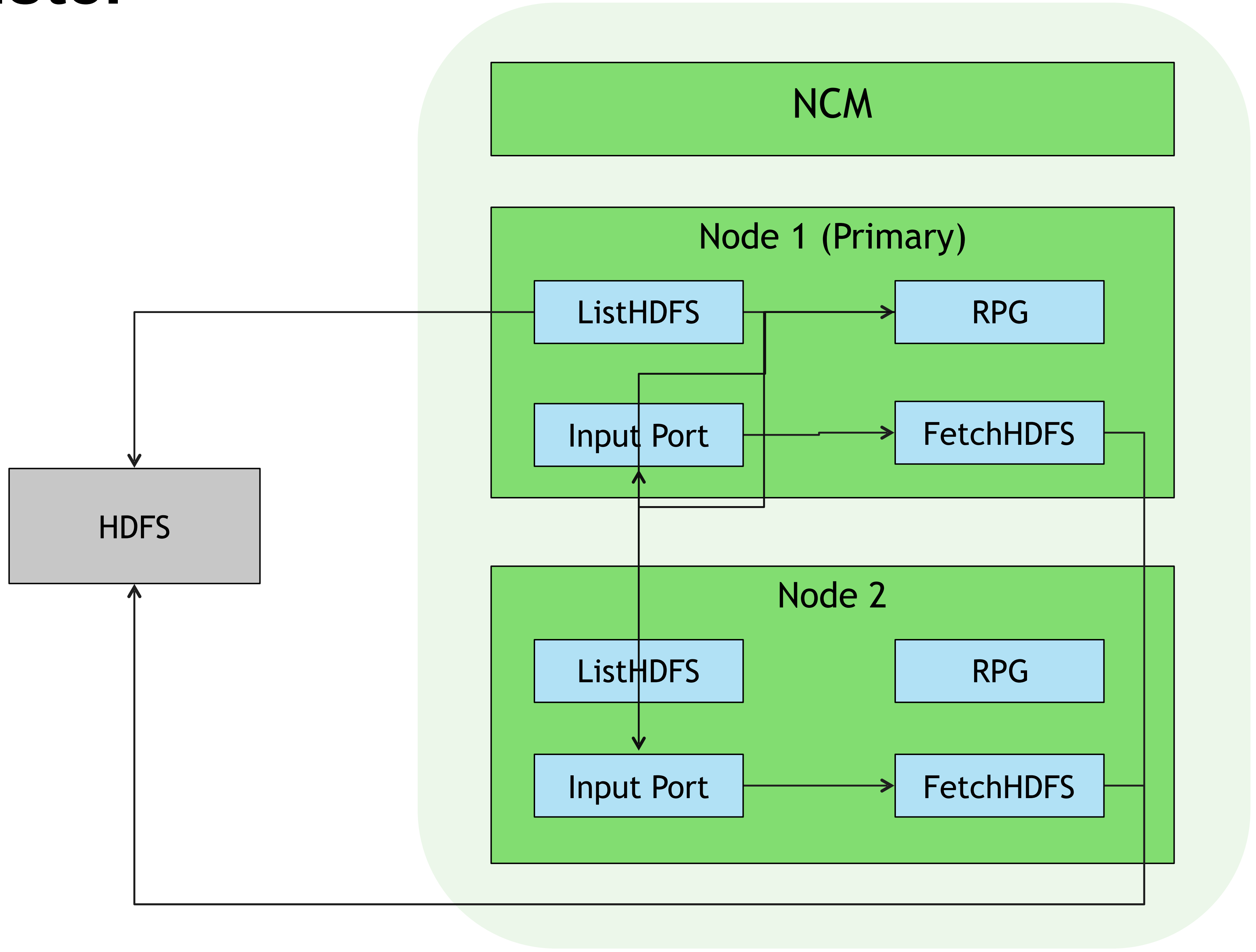
In	6 / 0 bytes	(5 min)
Read/Write	0 bytes / 731 bytes	(5 min)
Out	6 / 731 bytes	(5 min)
Tasks/Time	6 / 00:00:00.297	(5 min)

Name comms.failure
Queued 0 / 0 bytes

Name success
Queued 0 / 0 bytes

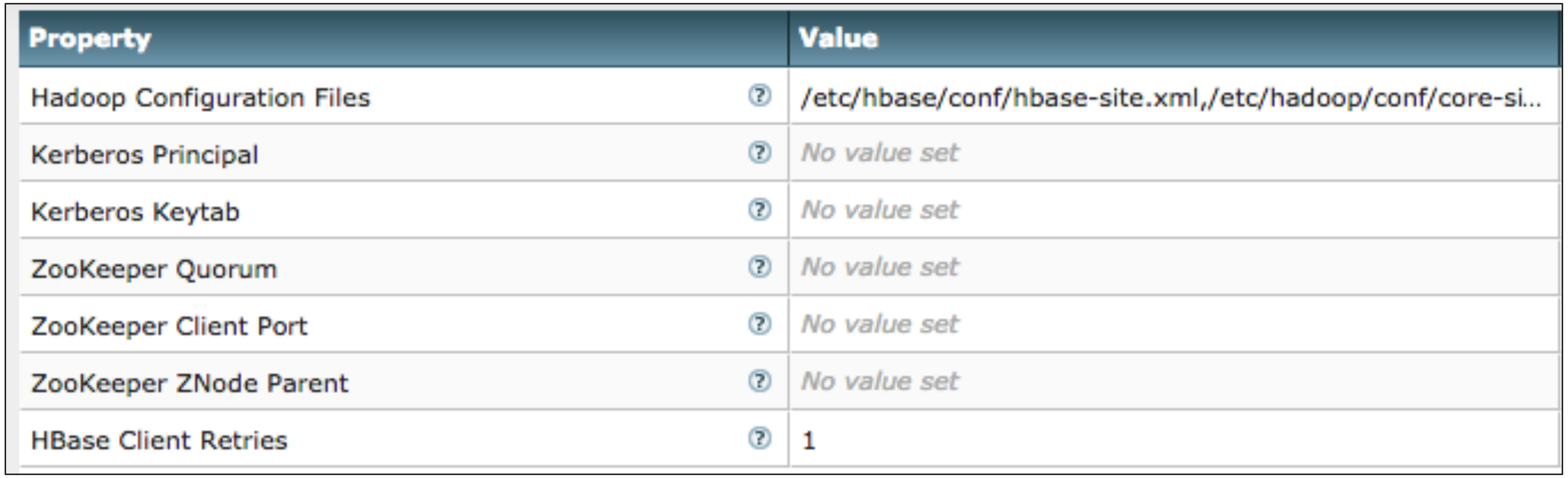
HDFS Retrieval in a Cluster

- Perform "list" operation on primary node
- Send results to Remote Process Group pointing back to same cluster
- Redistributes results to all nodes to perform "fetch" in parallel
- Same approach for ListFile + FetchFile and ListSFTP + FetchSFTP



HBase Integration

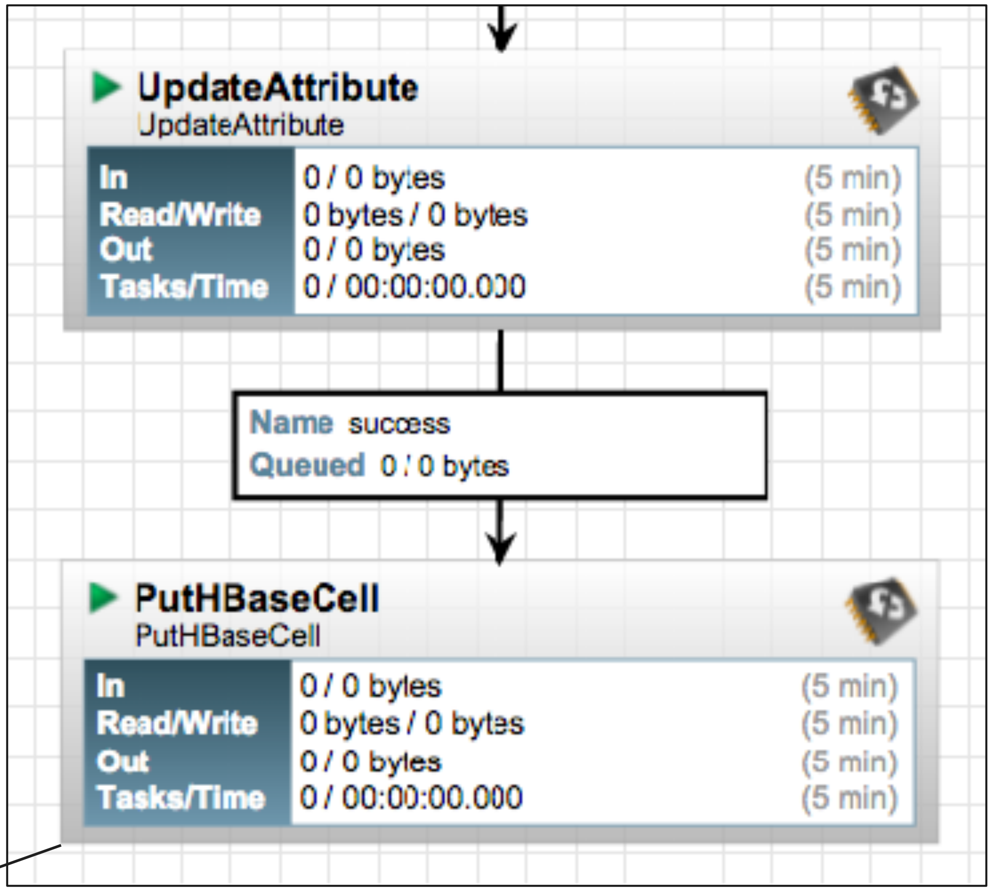
- ControllerService wrapping HBase Client
- Implementation provided for HBase 1.1.2 Client
- Other implementations could be built as an extension

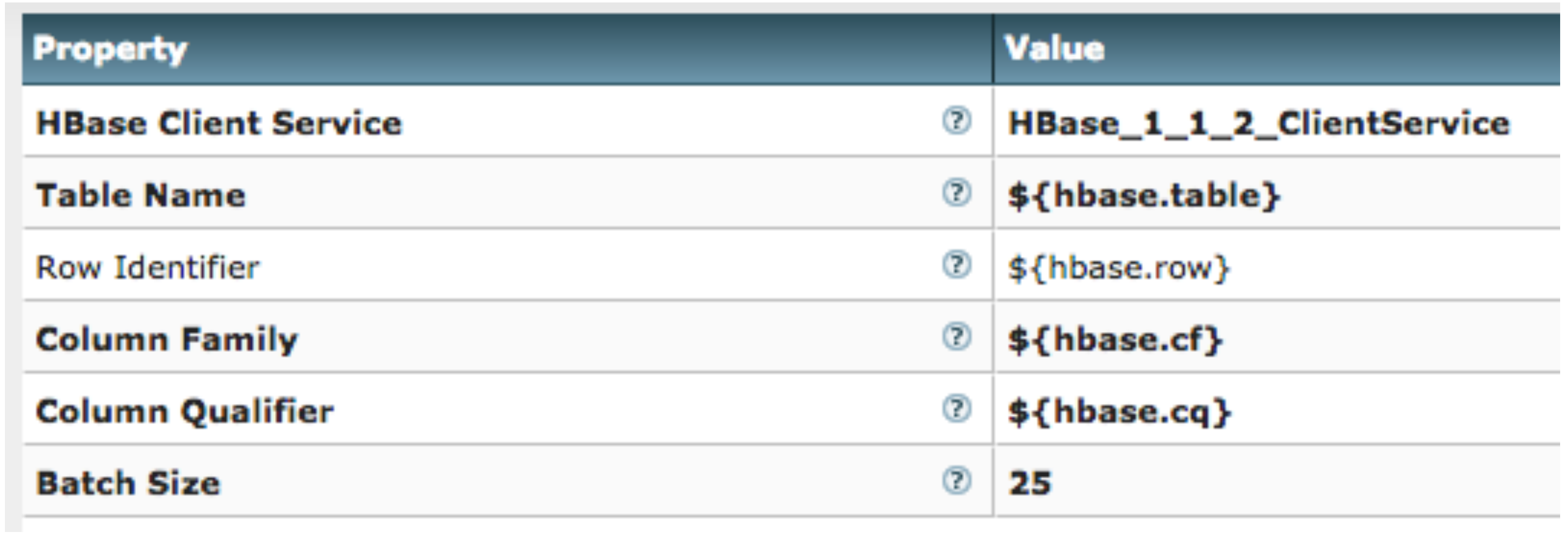


Property	Value
Hadoop Configuration Files	/etc/hbase/conf/hbase-site.xml,/etc/hadoop/conf/core-si...
Kerberos Principal	No value set
Kerberos Keytab	No value set
ZooKeeper Quorum	No value set
ZooKeeper Client Port	No value set
ZooKeeper ZNode Parent	No value set
HBase Client Retries	1

HBase Ingest – Single Cell

- Table, Row Id, Col Family, and Col Qualifier provided in processor, or dynamically from attributes
- FlowFile content becomes the cell value
- Batch Size to specify maximum number of cells for a single ‘put’ operation

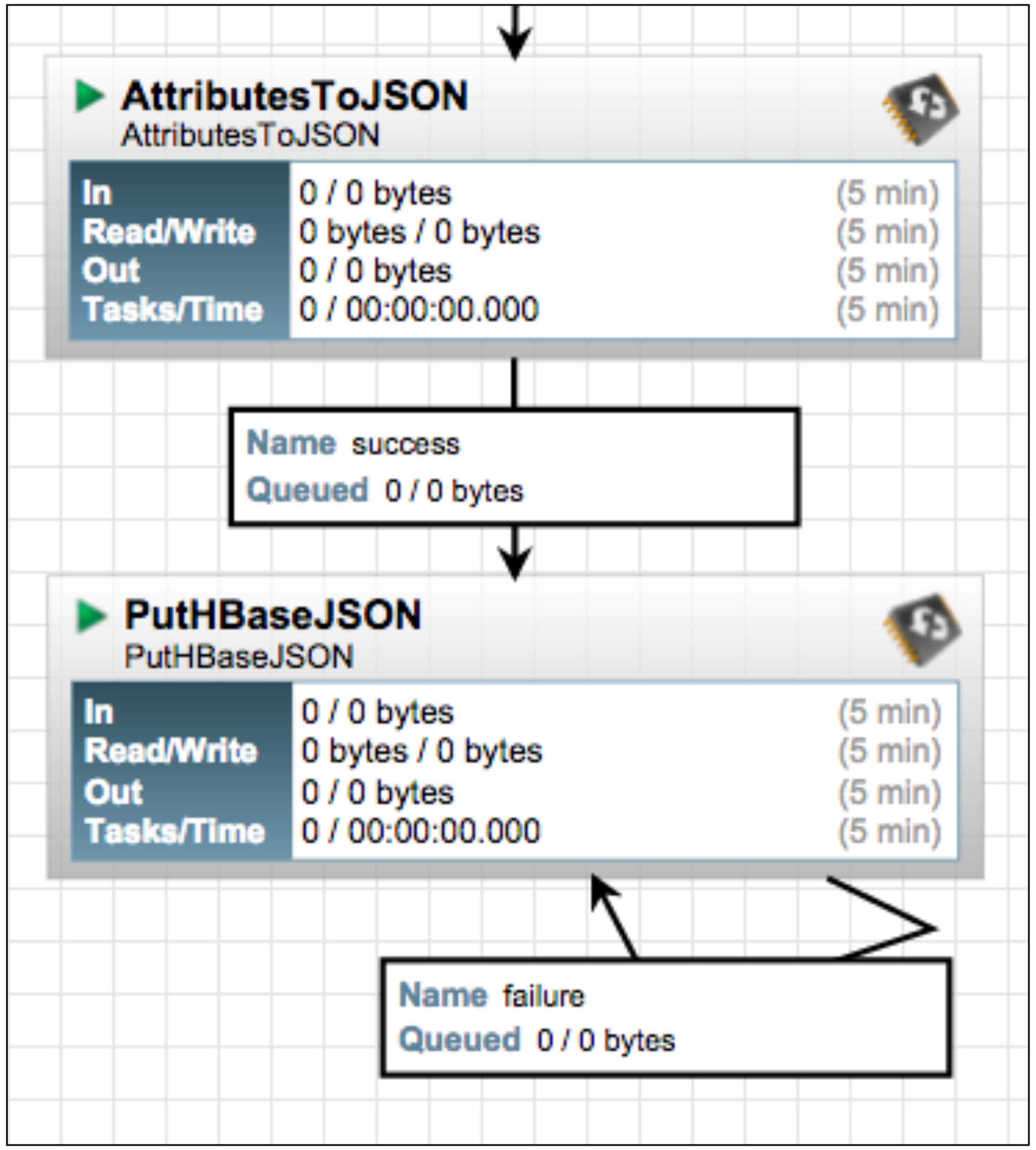




Property	Value
HBase Client Service	HBase_1_1_2_ClientService
Table Name	${hbase.table}
Row Identifier	${hbase.row}
Column Family	${hbase.cf}
Column Qualifier	${hbase.cq}
Batch Size	25

HBase Ingest – Full Row

- Table and Column Family provided in processor, or dynamically from attributes
- Row ID can be a field in JSON, or a FlowFile attribute
- JSON Field/Values become Column Qualifiers and Values
- Complex Field Strategy
  - Fail
  - Warn
  - Ignore
  - Text



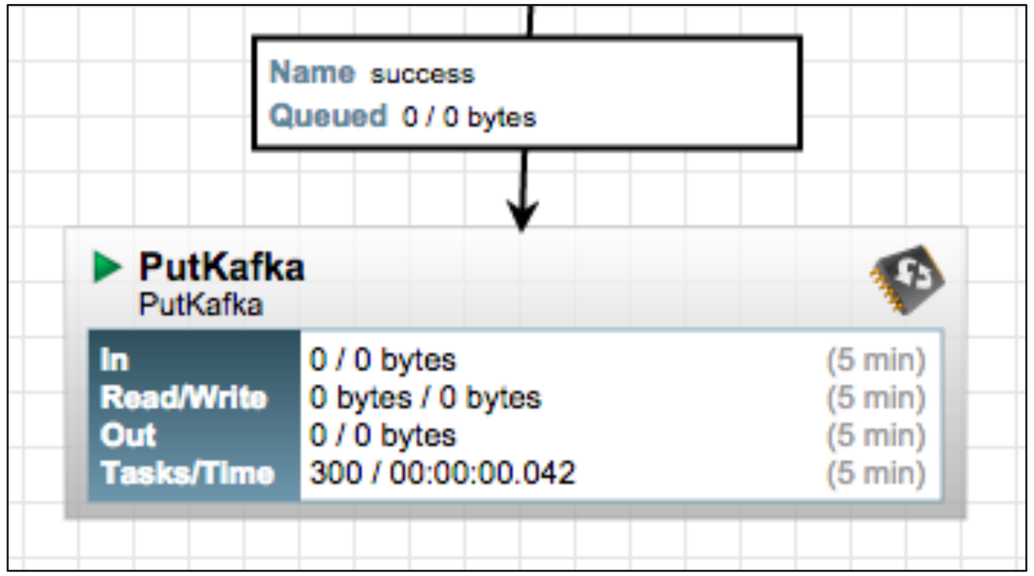
Kafka Integration

PutKafka

- Provide Broker and Topic Name
- Publishes FlowFile content as one or more messages
- Ability to send large delimited content, split into messages by NiFi

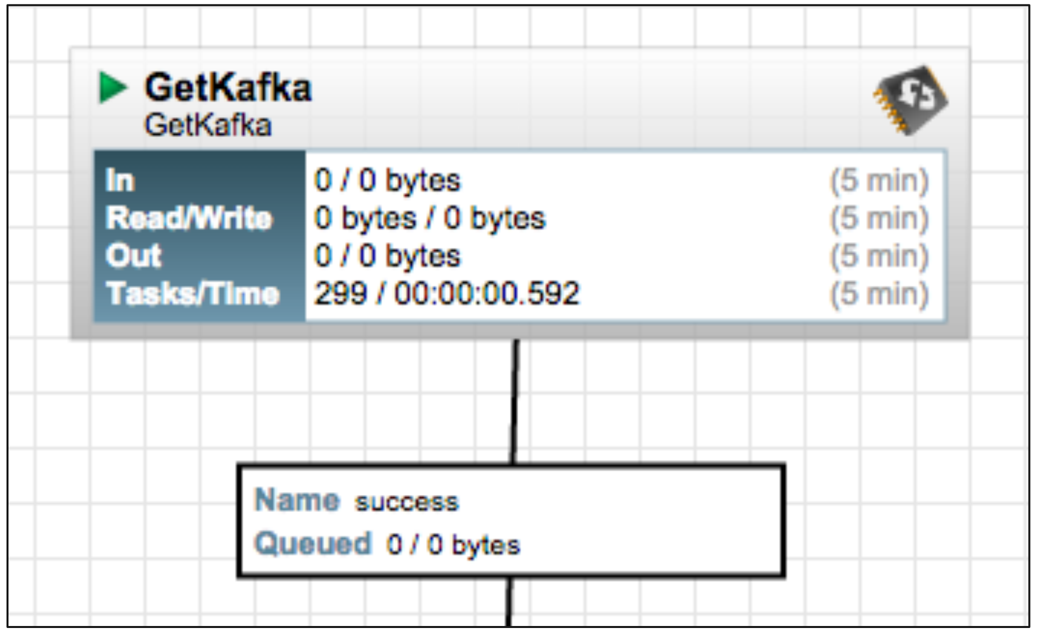
GetKafka

- Provide ZK Connection String and Topic Name
- Produces a FlowFile for each message consumed


Name success
Queued 0 / 0 bytes

PutKafka
PutKafka

In	0 / 0 bytes	(5 min)
Read/Write	0 bytes / 0 bytes	(5 min)
Out	0 / 0 bytes	(5 min)
Tasks/Time	300 / 00:00:00.042	(5 min)


GetKafka
GetKafka

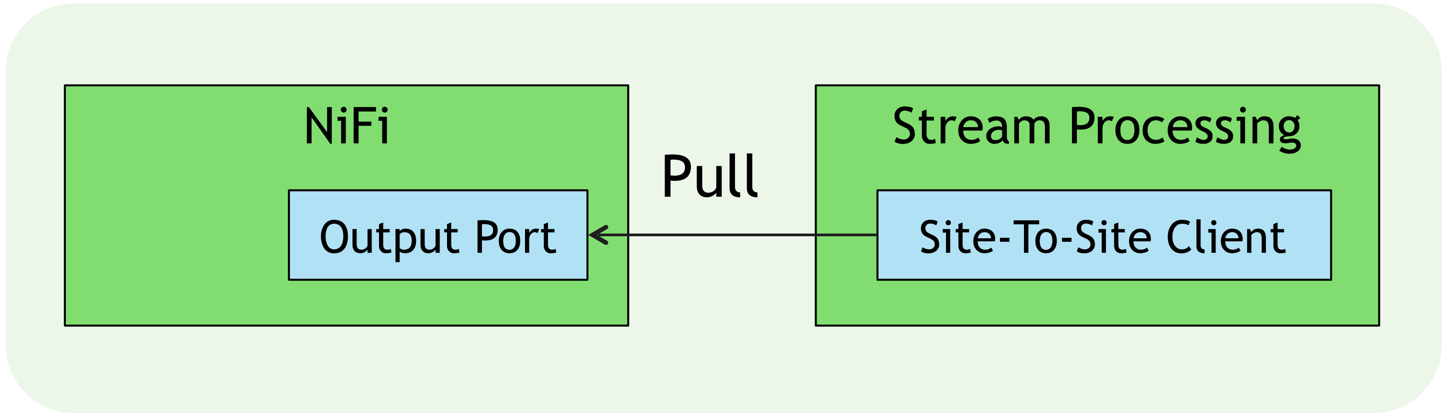
In	0 / 0 bytes	(5 min)
Read/Write	0 bytes / 0 bytes	(5 min)
Out	0 / 0 bytes	(5 min)
Tasks/Time	299 / 00:00:00.592	(5 min)

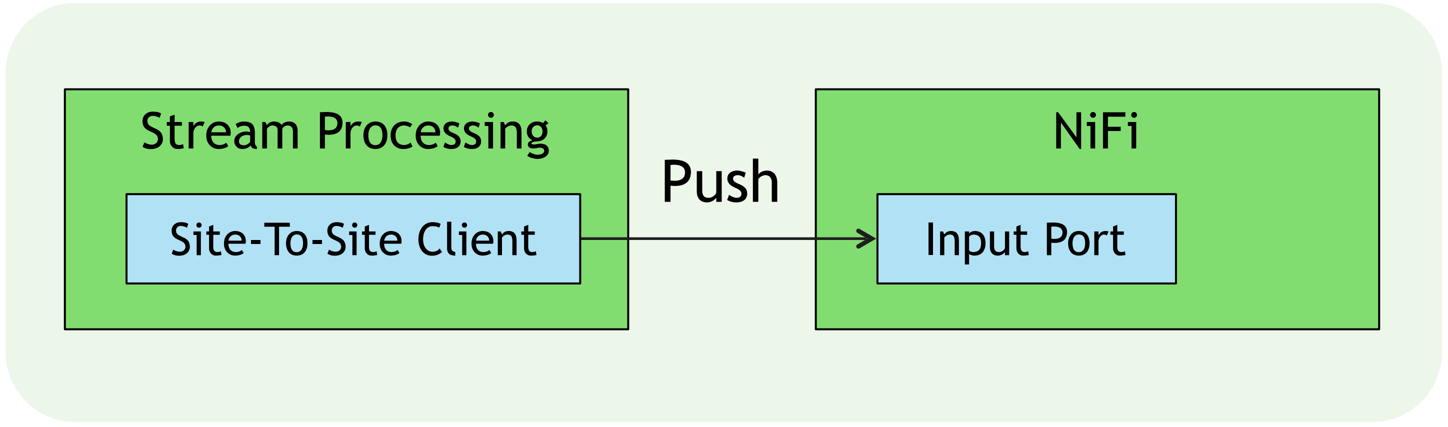
Name success
Queued 0 / 0 bytes

Stream Processing Integration

- Stream processing systems generally pull data, then push results
- NiFi Site-To-Site pushes and pulls between NiFi instances
- The Site-To-Site Client can be used from a stream processing platform

https://github.com/apache/nifi/tree/master/nifi-commons/nifi-site-to-site-client





Site-to-Site Client Overview

- Push to Input Port, or Pull from Output Port
- Communicate with NiFi clusters, or standalone instances
- Handles load balancing and reliable delivery
- Secure connections using certificates (optional)

```
SiteToSiteClientConfig clientConfig =
   new SiteToSiteClient.Builder()
      .url(“http://localhost:8080/nifi”)
      .portName(”My Port”)
      .buildConfig();
```

Site-To-Site Client Pulling

```
SiteToSiteClient client = ...


Transaction transaction =
client.createTransaction(TransferDirection.RECEIVE);
DataPacket dataPacket = transaction.receive();

while (dataPacket != null) {
...
}
transaction.confirm();

transaction.complete();
```

Site-To-Site Client Pushing

```
SiteToSiteClient client = ...


Transaction transaction =
client.createTransaction(TransferDirection.SEND);
NiFiDataPacket data = ...

transaction.send(data.getContent(), data.getAttributes());
transaction.confirm();

transaction.complete();
```

Current Stream Processing Integrations

Spark Streaming - NiFi Spark Receiver

- https://github.com/apache/nifi/tree/master/nifi-external/nifi-spark-receiver

Storm – NiFi Spout

- https://github.com/apache/nifi/tree/master/nifi-external/nifi-storm-spout

Flink – NiFi Source & Sink

- https://github.com/apache/flink/tree/master/flink-streaming-connectors/flink-connector-nifi

Apex - NiFi Input Operators & Output Operators

- https://github.com/apache/incubator-apex-malhar/tree/master/contrib/src/main/java/com/datatorrent/contrib/nifi

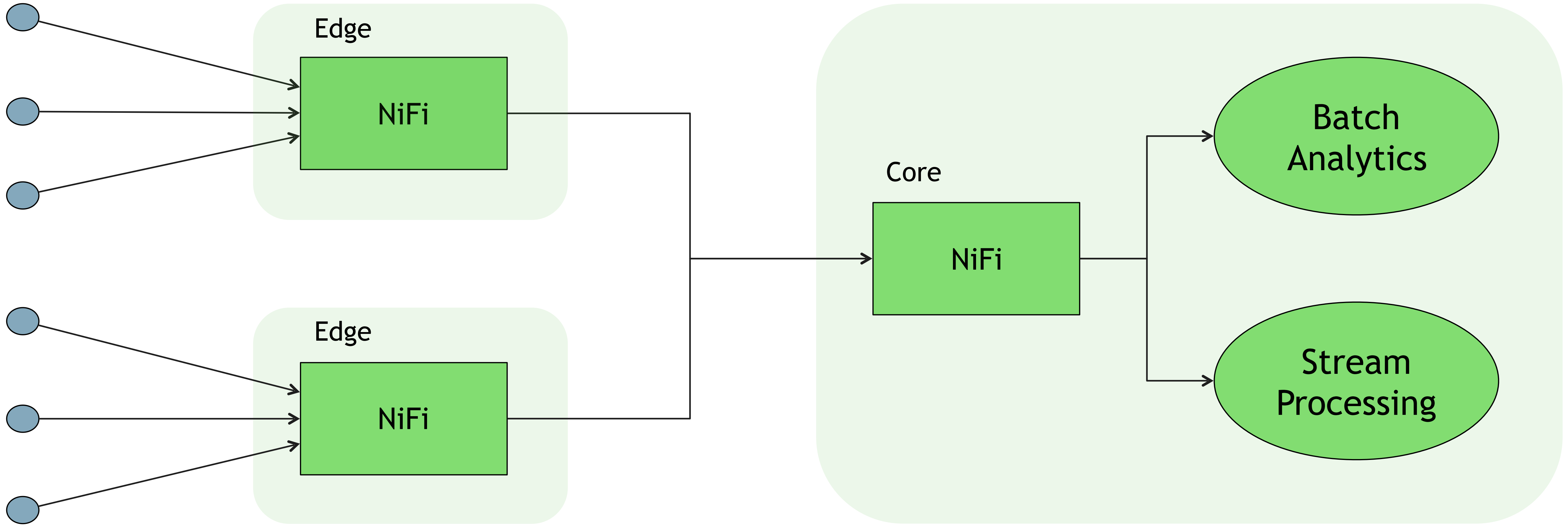
Other Relevant Integrations

- GetSolr, PutSolrContentStream
- FetchElasticSearch, PutElasticSearch
- GetMongo, PutMongo
- QueryCassandra, PutCassandraQL
- GetCouchbaseKey, PutCouchbaseKey
- QueryDatabaseTable, ExecuteSQL, PutSQL
- GetSplunk, PutSplunk
- And more! https://nifi.apache.org/docs.html

Use-Case/Architecture Discussion

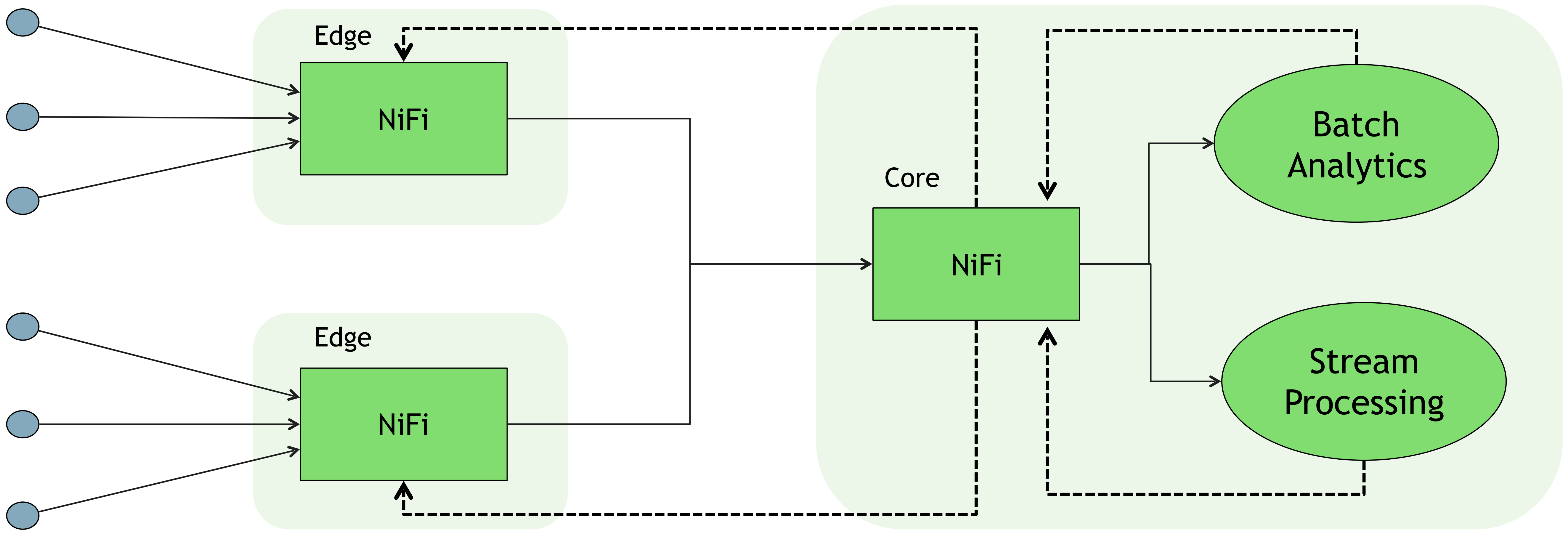
Drive Data to Core for Analysis

- Drive data from sources to central data center for analysis
- Tiered collection approach at various locations, think regional data centers



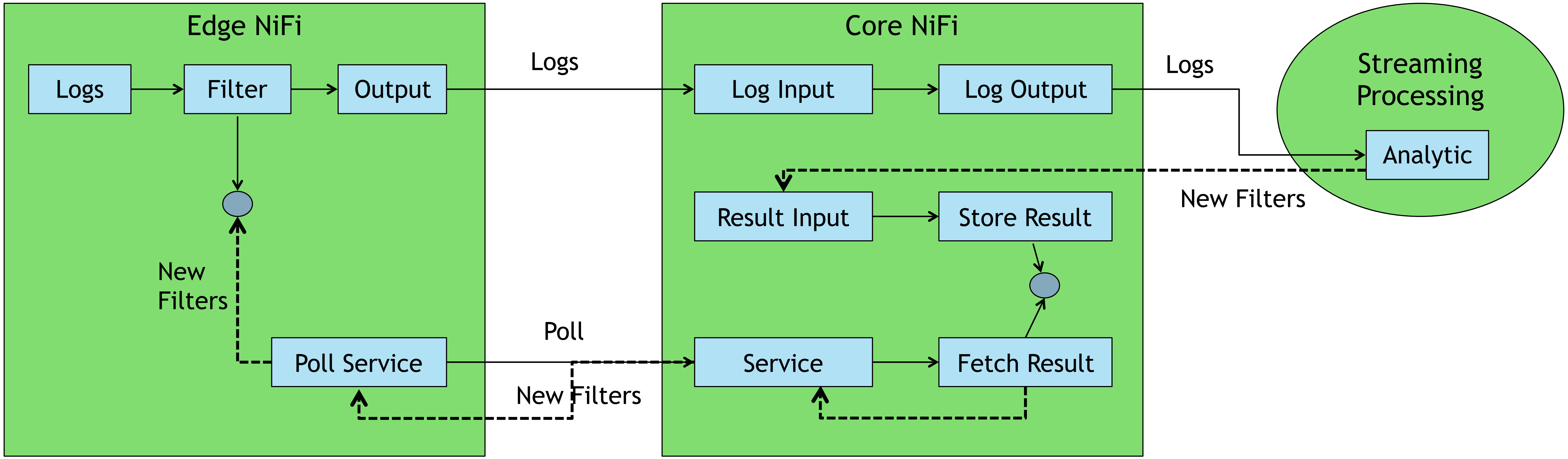
Dynamically Adjusting Data Flows

- Push analytic results back to core NiFi
- Push results back to edge locations/devices to change behavior

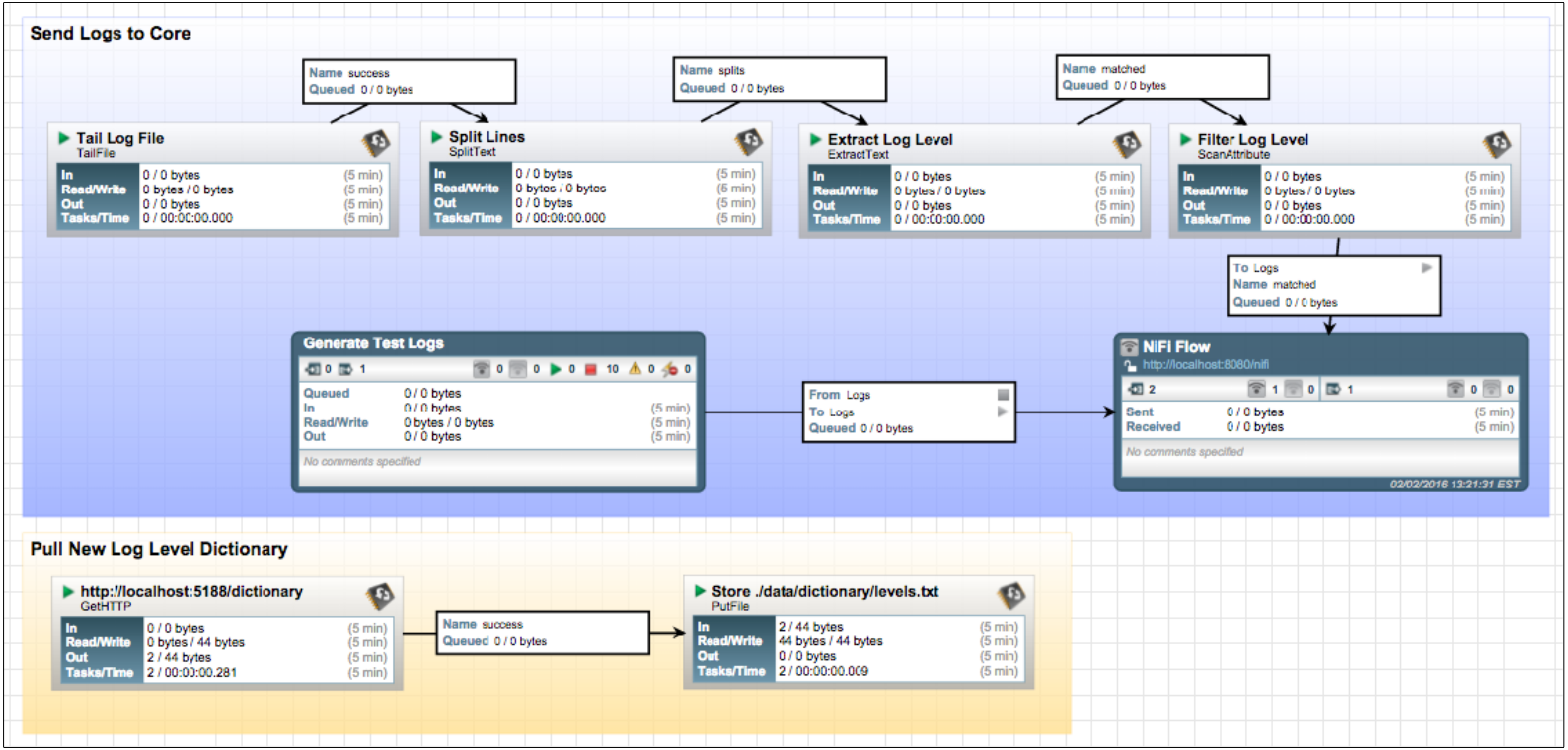


Example: Dynamic Log Collection

1. Logs filtered by level and sent from Edge -> Core
2. Stream processing produces new filters based on rate & sends back to core
3. Edge polls core for new filter levels & updates filtering



Dynamic Log Collection – Edge NiFi



Send Logs to Core

Name success
Queued 0 / 0 bytes

Name splits
Queued 0 / 0 bytes

Name matched
Queued 0 / 0 bytes

Tail Log File
TailFile

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)

Split Lines
SplitText

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)

Extract Log Level
ExtractText

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)

Filter Log Level
ScanAttribute

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)

To Logs
Name matched
Queued 0 / 0 bytes

Generate Test Logs
0 1 0 0 0 10 0 0

Queued	0 / 0 bytes	
In	0 / 0 bytes	(5 min)
Read/Write	0 bytes / 0 bytes	(5 min)
Out	0 / 0 bytes	(5 min)

No comments specified

From Logs
To Logs
Queued 0 / 0 bytes

NiFi Flow
http://localhost:8080/nifi
2 1 0 1 0 0

Sent	0 / 0 bytes	(5 min)
Received	0 / 0 bytes	(5 min)

No comments specified
02/02/2016 13:21:31 EST

Pull New Log Level Dictionary

http://localhost:5188/dictionary
GetHTTP

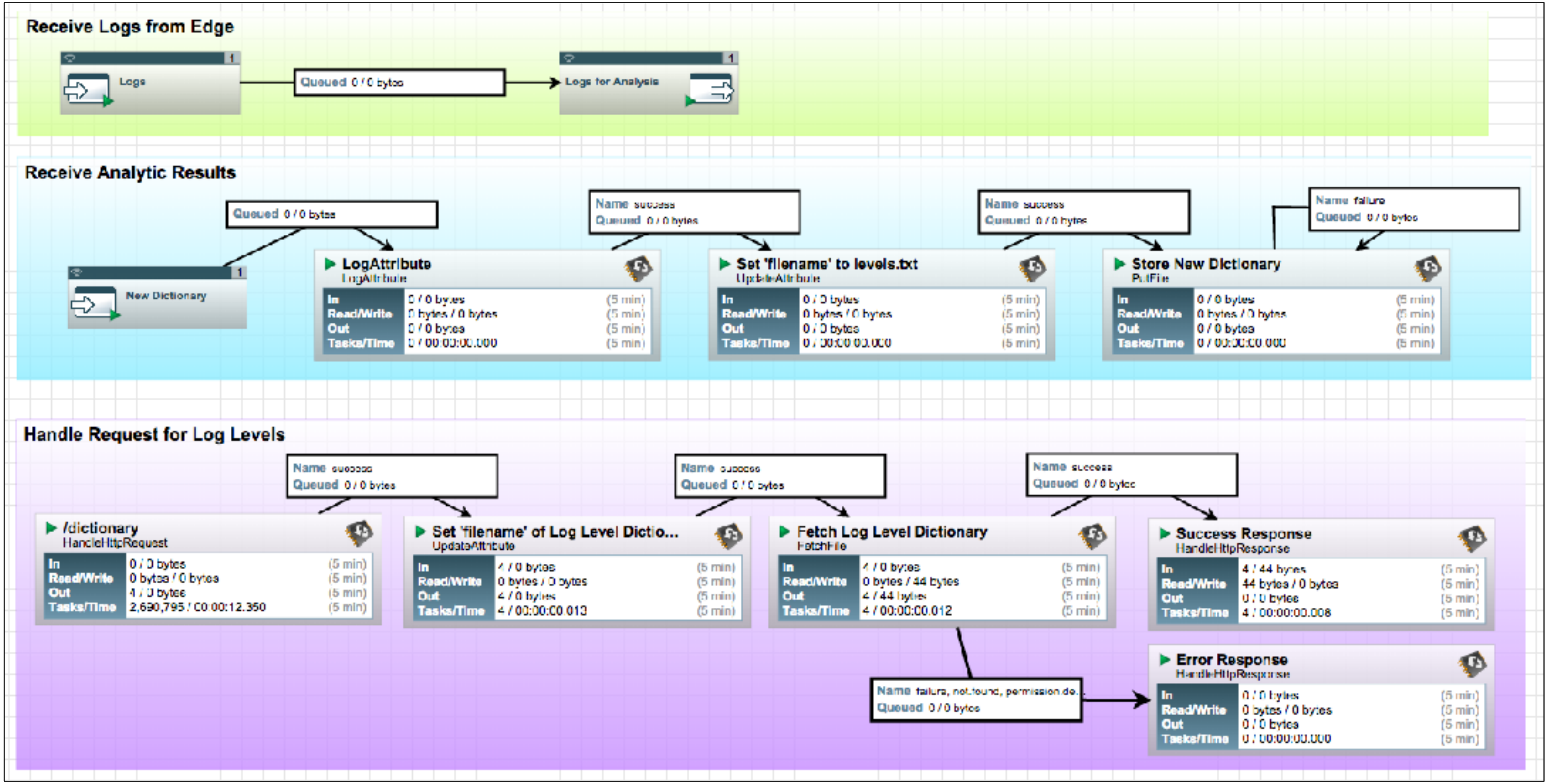
In	0 / 0 bytes	(5 min)
Read/Write	0 bytes / 44 bytes	(5 min)
Out	2 / 44 bytes	(5 min)
Tasks/Time	2 / 00:00:00.281	(5 min)

Name success
Queued 0 / 0 bytes

Store ./data/dictionary/levels.txt
PutFile

In	2 / 44 bytes	(5 min)
Read/Write	44 bytes / 44 bytes	(5 min)
Out	0 / 0 bytes	(5 min)
Tasks/Time	2 / 00:00:00.009	(5 min)

Dynamic Log Collection – Core NiFi



LogAttribute
LogAttribute

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)

Set 'filename' to levels.txt
UpdateAttribute

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)

Store New Dictionary
PutFile

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)

/dictionary
HandleHttpRequest

In	0 / 0 bytes	(5 min)
Read/Write	0 bytes / 0 bytes	(5 min)
Out	4 / 0 bytes	(5 min)
Tasks/Time	2,690,795 / 00:00:12.350	(5 min)

Set 'filename' of Log Level Dictio...
UpdateAttribute

In	4 / 0 bytes	(5 min)
Read/Write	0 bytes / 0 bytes	(5 min)
Out	4 / 0 bytes	(5 min)
Tasks/Time	4 / 00:00:00.013	(5 min)

Fetch Log Level Dictionary
FetchFile

In	4 / 0 bytes	(5 min)
Read/Write	0 bytes / 44 bytes	(5 min)
Out	4 / 44 bytes	(5 min)
Tasks/Time	4 / 00:00:00.012	(5 min)

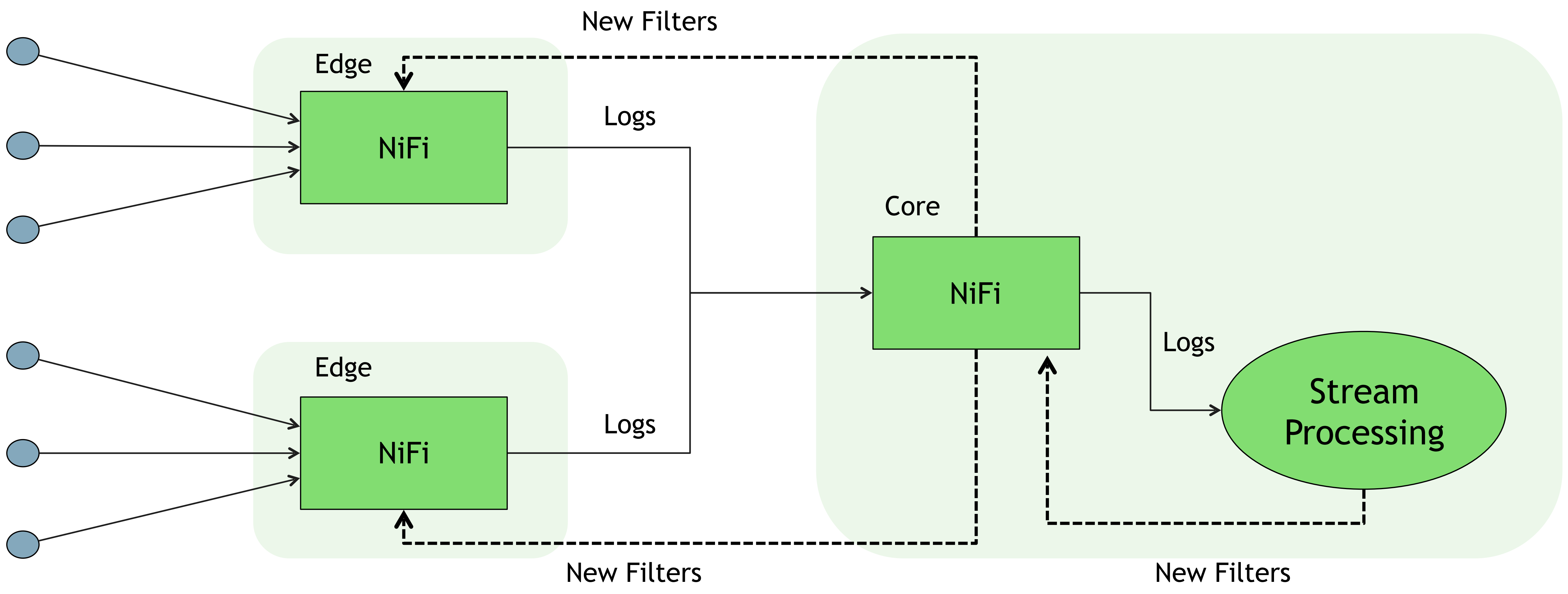
Success Response
HandleHttpResponse

In	4 / 44 bytes	(5 min)
Read/Write	44 bytes / 0 bytes	(5 min)
Out	0 / 0 bytes	(5 min)
Tasks/Time	4 / 00:00:00.008	(5 min)

Error Response
HandleHttpResponse

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)

Dynamic Log Collection Summary



The Future – Ecosystem Integrations

- Ambari
  - Support a fully managed NiFi Cluster through Ambari
  - Monitoring, management, upgrades, etc.
- Ranger
  - Ability to delegate authorization decisions to Ranger
  - Manage authorization controls through Ranger
- Atlas
  - Track lineage from the source to destination
  - Apply tags to data as its acquired

The Future – Apache NiFi

- HA Control Plane
  - Zero Master cluster, Web UI accessible from any node
  - Auto-Election of “Cluster Coordinator” and “Primary Node” through ZooKeeper
- HA Data Plane
  - Ability to replicate data across nodes in a cluster
- Multi-Tenancy
  - Restrict Access to portions of a flow
  - Allow people/groups with in an organization to only access their portions of the flow
- Extension Registry
  - Create a central repository of NARs and Templates
  - Move most NARs out of Apache NiFi distribution, ship with a minimal set

The Future – Apache NiFi

- Variable Registry
  - Define environment specific variables through the UI, reference through EL
  - Make templates more portable across environments/instances
- Redesign of User Interface
  - Modernize look & feel, improve usability, support multi-tenancy
- Continued Development of Integration Points
  - New processors added continuously!
- MiNiFi
  - Complimentary data collection agent to NiFi’s current approach
  - Small, lightweight, centrally managed agent that integrates with NiFi for follow-on dataflow management

Thank you