Installation 3

Planning your deployment

Date of Publish: 2020-04-28

https://docs.cloudera.com/

Contents

Deployment Scenarios

Your deployment scenario for installing, configuring, or upgrading your Hortonworks DataFlow (HDF) components depends on your particular use case.

Table 1: Installation Scenarios

Scenario	Installation Scenario	Steps
Installing an HDF Cluster	This scenario applies if you want to install the entire HDF platform, consisting of all flow management and stream processing components on a new cluster. The stream processing components include the new Streaming Analytics Manager (SAM) modules that are in GA (General Availability). This includes the SAM Stream Builder and Stream Operations modules but does not include installing the technical preview version of SAM Stream Insight, which is powered by Druid and Superset. This scenario requires that you install an HDF cluster.	1. Install Ambari. 2. Install databases. 3. Install the HDF management pack. 4. Install an HDF cluster using Ambari.
Installing HDF Services on a New HDP Cluster	This scenario applies to you if you are both an Hortonworks Data Platform (HDP) and HDF customer and you want to install a fresh cluster of HDP and add HDF services. The stream processing components include the new (SAM) and all of its modules. This includes installing the technical preview version of the SAM Stream Insight module, which is powered by Druid and Apache Superset. This scenario requires that you install both an HDF cluster and an HDP cluster.	1. Install Ambari. 2. Install databases. 3. Install an HDP cluster using Ambari. 4. Install the HDF management pack. 5. Update the HDF base URL. 6. Add HDF services to an HDP cluster
Installing HDF Services on an Existing HDP Cluster	You have an existing HDP cluster with Apache Storm and or Apache Kafka services and want to install Apache NiFi or NiFi Registry modules on that cluster. This requires that you upgrade to the latest version of Apache Ambari and HDP, and then use Ambari to add HDF services to the upgraded HDP cluster.	1. Upgrade Ambari 2. Upgrade HDP 3. Install Databases 4. Install HDF Management Pack 5. Update HDF Base URL 6. Add HDF Services to HDP cluster

Table 2: Upgrade Scenarios

Scenario	Upgrade Scenario	Documentation
Upgrading an HDF-only cluster	You have an existing Ambari-managed HDF cluster and want to upgrade it using an Express upgrade.	Resources for each upgrade scenario are available in the Ambari-Managed HDF Upgrade Guide.
Upgrading HDF 3.4.x services on an HDP cluster.	You have an existing Ambari-managed HDP cluster with HDF 3.4.x services installed. In the HDF 3.4.x release, NiFi and NiFi Registry were available to install on HDP. You want to upgrade your HDP cluster to HDP 3.1.5 and you want to upgrade your HDF services to the services available with HDF 3.5.1.	
Upgrading an HDP cluster with HDF 3.3.x	You have an existing Ambari-managed HDP cluster with HDF 3.3.x services installed. In the HDF 3.3.x release, NiFi, SAM, and Schema Registry were available to install on HDP.	

HDF Cluster Types and Recommendations

Cluster Type	Description	Number of VMs or Nodes	Node Specification	Network
Single VM HDF Sandbox	Evaluate HDF on local machine. Not recommended to deploy anything but simple applications.	1 VM	At least 4 GB RAM	
Evaluation Cluster	Evaluate HDF in a clustered environment. Used to evaluate HDF for simple data flows and streaming applications.	3 VMs or no des	• 16 GB of RAM • 8 cores/vCores	
Small Development Cluster	Use this cluster in development environments.	6 VM s/No des	• 16 GB of RAM • 8 cores or vCores	
Medium QE Cluster	Use this cluster in QE environments.	8 VMs/Nodes	• 32 GB of RAM • 8 to16 cores or vCores	
Small Production Cluster	Use this cluster in small production environments.	15 VMs/Nodes	• 64 - 128 GB of RAM • 8 - 16 cores of RAM	1 GB Bonded Nic
Medium Production Cluster	Use this cluster in a medium production environment.	24 VMs/Nodes	• 64 - 128 GB of RAM • 8 - 16 cores of RAM	10 GB bonded network interface card (NIC)
Large Production Cluster	Use this cluster in a large production environment.	32 VMs/Nodes	• 64 - 128 GB of RAM • 16 cores of RAM	10 GB Bonded NIC

More Information

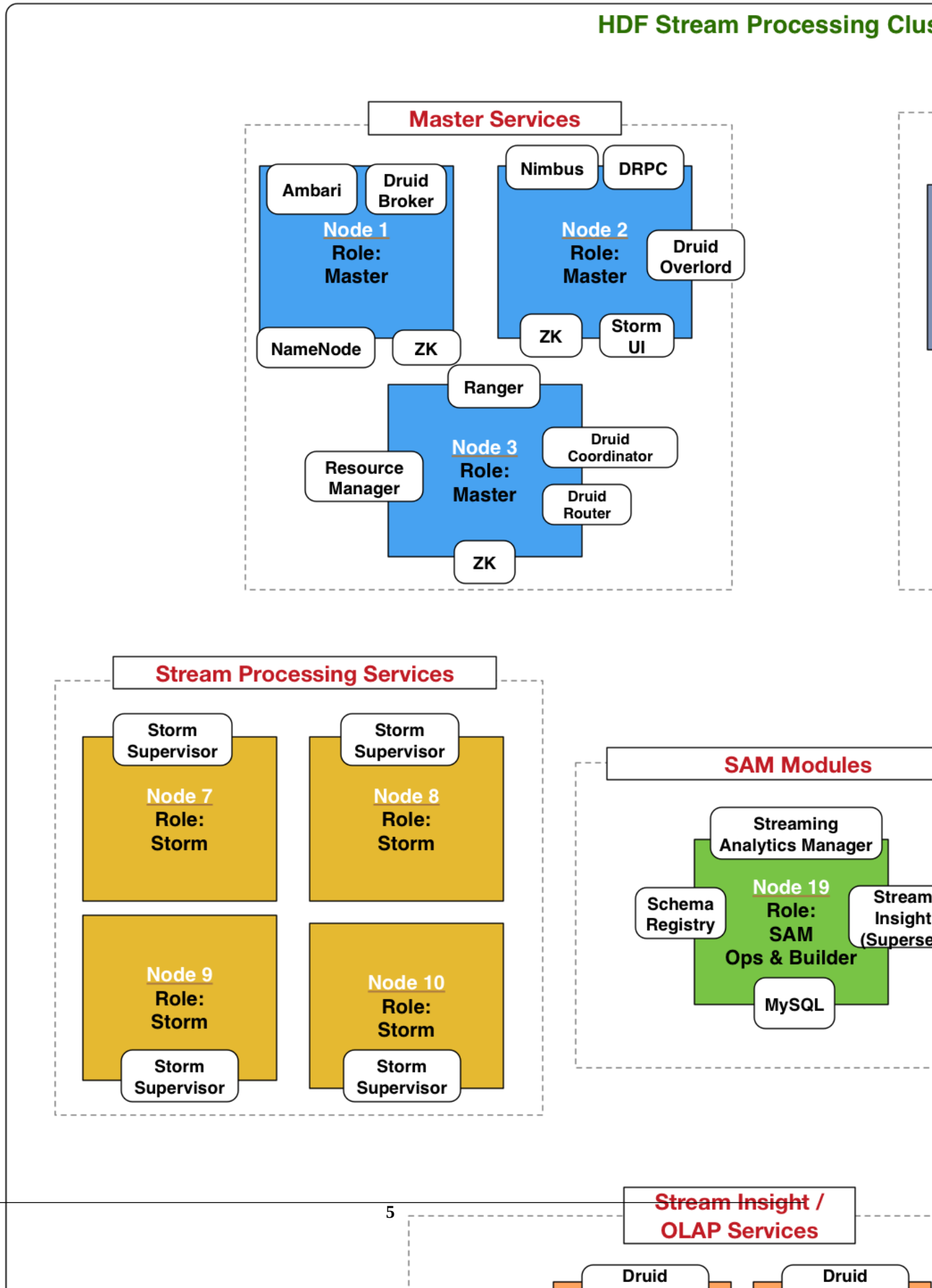
Download the Sandbox

Production Cluster Guidelines

General guidelines for production guidelines for service distribution:

- NiFi, Storm, and Kafka should not be located on the same node or virtual machine.
- NiFi, Storm, and Kafka must have a dedicated ZooKeeper cluster with at least three nodes.
- If the HDF SAM is being used in an HDP cluster, the SAM should not be installed on the same node as the Storm worker node.

The following diagram illustrates how services could be distributed for a small production cluster across 19 nodes:


HDF Stream Processing Clus
Master Services
Ambari
Druid Broker
Node 1
Role:
Master
NameNode
ZK
Nimbus
DRPC
Node 2
Role:
Master
Druid Overlord
ZK
Storm UI
Ranger
Node 3
Role:
Master
Resource Manager
Druid Coordinator
Druid Router
ZK
Stream Processing Services
Storm Supervisor
Node 7
Role:
Storm
Storm Supervisor
Node 8
Role:
Storm
Node 9
Role:
Storm
Storm Supervisor
Node 10
Role:
Storm
Storm Supervisor
SAM Modules
Streaming Analytics Manager
Schema Registry
Node 19
Role:
SAM
Ops & Builder
Stream Insight
MySQL
Stream Insight / OLAP Services
Druid
Druid

Hardware Sizing Recommendations

Recommendations for Kafka

- Kafka Broker node: eight cores, 64 GB to128 GB of RAM, two or more 8-TB SAS/SSD disks, and a 10- GbE NIC.
- Minimum of three Kafka broker nodes
- Hardware Profile: More RAM and faster speed disks are better; 10 GbE NIC is ideal.
- 75 MB per sec per node is a conservative estimate. You can go much higher if more RAM and reduced lag between writing/reading and therefore 10 GB NIC is required.

With a minimum of 3 nodes in your cluster, you can expect 225 MB/sec data transfer.

You can perform additional further sizing by using the following formula: num_brokers = desired_throughput (MB/sec) / 75

Recommendations for Storm

- Storm Worker Node: 8 core, 64 GB RAM, 1 GbE NIC
- Minimum of 3 Storm worker nodes
- Nimbus Node: Minimum 2 Nimbus nodes , 4 core, 8 GB RAM
- Hardware profile: disk I/O is not that important; more cores are better.
- 50 MB per sec per node with low to moderate complexity topology reading from Kafka and no external lookups . Medium-complexity and high-complexity topologies might have reduced throughput.

With a minimum 2 nimbus, 2 worker cluster, you can expect to run 100 MB/sec of low to medium complexity topology.

Further sizing can be done as follows. Formula: num_worker_nodes = desired_throughput(MB/sec) / 50

Sizing your Flow Management cluster

Learn about the factors that impact the resource requirements and the performance of your Flow Management cluster.

When considering how to size and configure Flow Management clusters, it is important that you keep each of the following considerations in mind.

- Data flow design
- Principles of NiFi design
- Cluster layout
- Disk configuration
- Memory and CPU intensive processors

Data flow design

Learn about the flow design factors that impact your cluster sizing needs and the performance of your data flow.

Note:

Flow design is the most important factor influencing the expected data flow performance, as well as the type and amount of resources used by NiFi.

You can use NiFi for a wide array of use cases, and the resource requirements are greatly determined by data flow design. Depending on the actions it performs in a data flow, each processor may or may not be required to read or to write the processed data from or on disks.

For example, a flow ingesting 100 MB of data per second with its first processor, may need to read and write this data on disks multiple times before the result is sent to the final destination. If the data flow has four processors writing the content on disks before being sent to the final destination, the disks used for the content repositories in the NiFi cluster should be able to handle 400 MB per second at the cluster level.

NiFi design

Learn about the NiFi design decisions that impact Flow Management cluster sizing.

NiFi is designed to use all the available resources of the nodes where it is running. NiFi takes advantage of:

- All available cores
- All the network capacity
- All the disk speed and capacities

Note:

It is crucial to understand that NiFi data flow behavior depends on three major factors:

- data flow source
- flow operations during data transfer
- data flow target

For more information about principles involved in the NiFi design, see the *Apache NiFi Overview*.

Related Information

Apache NiFi Overview

Cluster layout

Learn about Flow Management cluster layout recommendations to optimize your flow management cluster in production environments.

At least three nodes are recommended for a Flow Management cluster used in production environments. NiFi must be running on dedicated nodes.

When NiFi is clustered, you should use an external ZooKeeper cluster. In production deployments, the ZooKeeper instances should not be co-located with the NiFi instances.

Disk configuration

Learn about NiFi disk configuration and how to optimize for the three repositories on disk.

On most modern systems, the disk throughput is lower than the network throughput, so the network is usually not a bottleneck. For most data movement use cases, the CPU usage is much lower than the disk I/O, but it is still important to monitor the CPU and tune the number of threads per processor. See *Tuning your Data Flow* for recommendations about fine tuning threads usage.

NiFi has three repositories on disk and the disk configuration is a very important performance factor:

- Content Repository
  - Contains the content of each FlowFile
  - Sequential disk I/O (ideally leveraging the OS cache)
- FlowFile (metadata) Repository
  - Contains the FlowFile attributes and current FlowFile state (which queue it is in) for each FlowFile
  - Sequential and random disk I/O
- Provenance (metadata) Repository

- Contains a provenance log with entries for every action performed on a FlowFile (merge, drop, and so on).
- Sequential and random disk I/O

Every FlowFile that NiFi receives or creates is immediately written to disk in the content repository for fault tolerance. Subsequent FlowFile content modifications (decompression, format conversion, and so on) are also written to the content repository. Processors that do not modify the content, such as RouteOnAttribute, do not impact the content repository. Instead, the FlowFile repository keeps a pointer for each FlowFile showing its state, such as which queue it is located in. This optimization eliminates the need for redundant writes to the content repository.

For higher performance, configure multiple disks for both the content and provenance repositories.

For more information, see *File System Content Repository Properties* and *Write Ahead Provenance Repository Properties* in the *NiFi System Properties documentation* and *Configuration Best Practices*.

Related Information

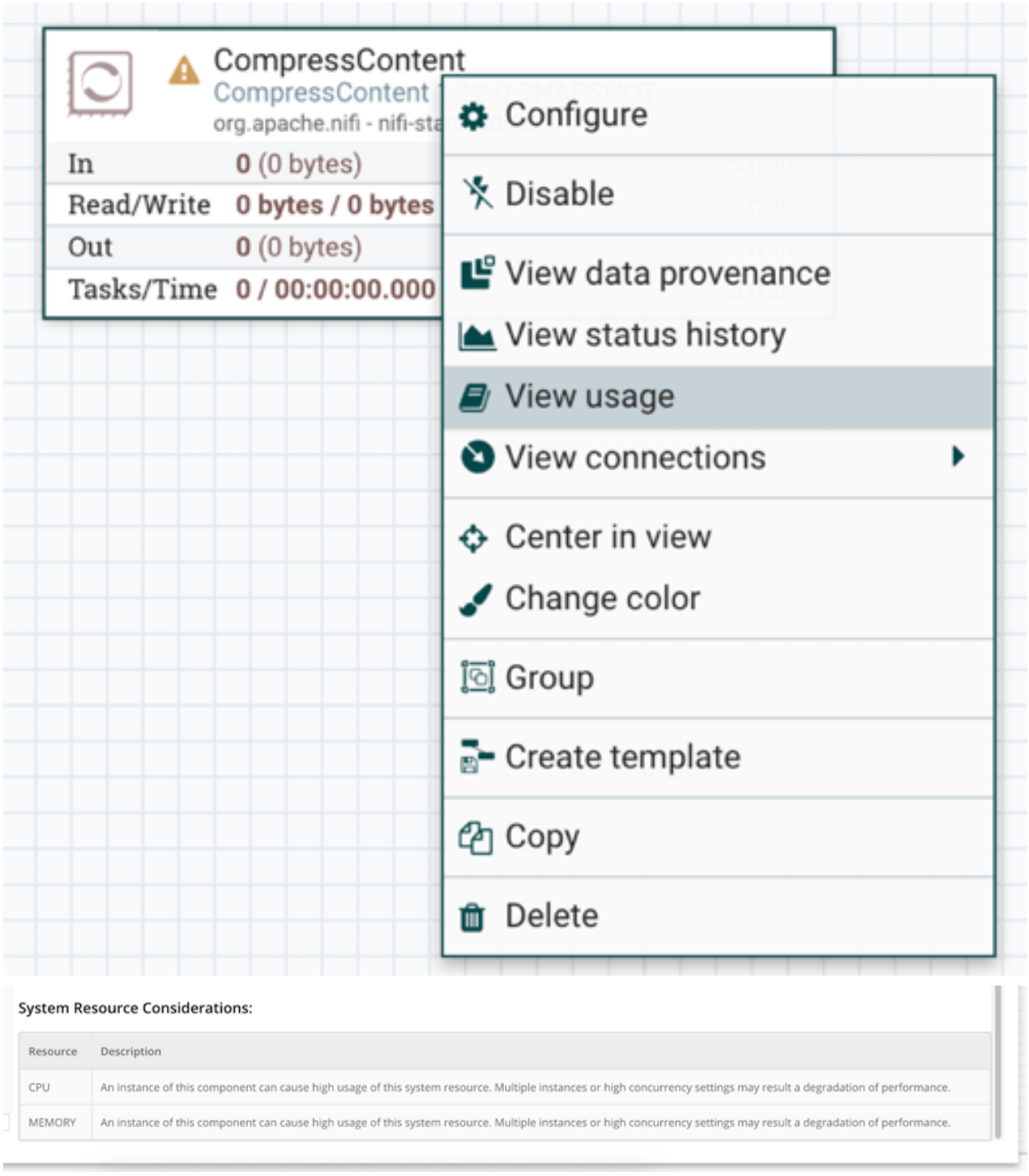
File System Content Repository Properties
Write Ahead Provenance Repository Properties
Configuration Best Practices
Tuning your Data Flow

Resource intensive processors

Learn how to identify resource intensive processors.

In the Apache NiFi documentation, you can review information about whether a processor is CPU intensive or memory intensive.

For example, if you right-click the CompressContentprocessor to open its documentation, you can see that this processor can be both memory and CPU intensive.



Recommendations

Learn how to configure your Flow Management cluster with sizing considerations in mind.

Cloudera recommends the following setup for on-premises, bare metal installations:

- 1 RAID 1 or 10 array for the OS
- 1 RAID 1 or 10 array for the FlowFile repository
- 1 or many RAID 1 or 10 array(s) for the content repository

- 1 or many RAID 1 or 10 array(s) for the provenance repository

For high performance setup, Cloudera recommends SSDs over spinning disks.

For cloud environments, larger disks usually provide better throughputs. Review your cloud provider documentation for more information.

In terms of memory, NiFi is optimized to support FlowFiles of any size. This is achieved by never materializing the file into memory directly. Instead, NiFi uses input and output streams to process events (there are a few exceptions with some specific processors). This means that NiFi does not require significant memory even if it is processing very large files. Most of the memory on the system should be left available for the OS cache. By having a large enough OS cache, many of the disk reads are skipped completely. Consequently, unless NiFi is used for very specific memory oriented data flows, setting the Java heap to 8 GB or 16 GB is usually sufficient.

The performance you can expect directly depends on the hardware and the flow design. For example, when reading compressed data from a cloud object store, decompressing the data, filtering it based on specific values, compressing the filtered data, and sending it to a cloud object store, you can achieve the following results:

Nodes	Data rate per second	Events per second	Data rate per day	Events per day
1	192.5 MB	946,000	16.6 TB	81.7 billion
5	881 MB	4.97 million	76 TB	429.4 billion
25	5.8 GB	26 million	501 TB	2.25 trillion
100	22 GB	90 million	1.9 PB	7.8 trillion
150	32.6 GB	141.3 million	2.75 PB	12.2 trillion

Data rates and event rates were captured running the flow described above on Google Kubernetes Engine. Each node has 32 cores, 15 GB RAM, and a 2 GB heap. The Content Repository is a 1 TB Persistent SSD (400 MB per second write, 1200 MB second read).

NiFi scales well, both vertically and horizontally. Depending on the number of data flows running in the NiFi cluster and your operational requirements, you can add nodes to the NiFi cluster over time to meet your needs.

With this information in mind, Cloudera recommends:

- At least 4 cores per NiFi node (more is better and 8 cores usually provides the best starting point for the most common use cases)
- At least 6 disks per NiFi node to ensure dedicated disks for repositories
- At least 4GB of RAM for the NiFi heap

Now that you have finished reviewing the Flow Management cluster sizing considerations, see *Processing one billion events per second with NiFi* for additional information and a use case walk through.

Related Information

Processing one billion events per second with Apache NiFi