Jared Gray

16 October 2019

Contents

[1. Introduction 1](#_Toc22165640)

[2. WAN Ingestion Service 1](#_Toc22165641)

[2.1. Creating a New WAN Ingestion Job 2](#_Toc22165644)

[2.2. Executing a WAN Ingestion Job 3](#_Toc22165645)

[2.3. Describing a Single WAN Ingestion Job 3](#_Toc22165646)

[2.4. Listing WAN Ingestion Jobs 4](#_Toc22165647)

[2.5. Getting Data for a Chunk 5](#_Toc22165648)

[3. LAN Ingestion Service 6](#_Toc22165649)

[3.1. Creating a New LAN Ingestion Job 6](#_Toc22165651)

[3.2. Executing a LAN Ingestion Job 7](#_Toc22165652)

[3.3. Describing a Single LAN Ingestion Job 7](#_Toc22165653)

[3.4. Listing LAN Ingestion Jobs 8](#_Toc22165654)

[4. Use Case: 1 TB, Two Data Centers 9](#_Toc22165655)

# Introduction

We define an API herein for the bulk ingestion of data over a network. Our conceptual model involves a single external data source and a potentially large number of local sinks. It is assumed that copying data from the external source is slow but that throughput between local nodes is fast.

According to this API, the ingestion proceeds in two discrete phases. In the first phase, a (proportionally small) number of nodes copy the data from the remote source and ensure their copies are consistent with one another. During the second phase, we broadcast the data obtained from the previous phase to a (proportionally large) number of nodes within the local geographic site.

# WAN Ingestion Service

The Wide Area Network (WAN) Ingestion Service copies all data from a remote source onto the local storage of several on-site nodes. Ingested data are broken up into a series of labeled **chunks** for easier consumption by downstream services. Time-to-live mechanics are provided to free up disk space for future data.

Every data node that implements this service maintains a full copy of the remote data on local disk. It is assumed that the remote data source may be mutable. Supposing a mutation occurs in the data source following an ingestion process, the *WAN Ingestion Service* will not reflect the mutation.

Within the *WAN Ingestion Service*, all constituent service nodes maintain identical chunk data. If the data source is mutated during the ingestion process, the copies we store are consistent with one another (but not necessarily with the source). When we increase capacity for the WAN Ingestion Service, the new data nodes(s) must successfully ingest all live chunks before becoming available.



## Creating a New WAN Ingestion Job

With the CreateJob API, a user defines a new WAN ingestion job. Its input parameters are as follows:

| Name | Type | Description | Required? |
| --- | --- | --- | --- |
| sourceUri | String | URI for the data source | Yes |
| chunkSizeUnit | Enum | One of {KB, MB, GB, TB} | No |
| chunkSizeValue | Integer | Magnitude of the chunk size | No |
| description | String | Human-readable characterization of the job | No |
| timeToLiveUnit | Enum | One of {SECOND, MINUTE, HOUR, DAY} | No |
| timeToLiveValue | Integer | Magnitude of the time-to-live | No |

Table : Request Parameters for CreateJob

In the event of successful request creation (HTTP status code 201), the API returns an identifier for the job (jobId) that was created. This API may return a status code of 400 when input validation fails.

## Executing a WAN Ingestion Job

We define the ExecuteJob API to allow the user to execute an existing ingestion job. Its sole input parameter is a jobId. The output parameters are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required? |
| status | Enum | Execution status: {SUCCESS, FAILURE} | Yes |
| createdChunks | List [Chunk] | Chunks successfully created and replicated | Yes |
| failedChunks | List [Chunk] | Chunks created but unsuccessfully replicated | No |

Table : Response Parameters for ExecuteJob

Accordingly, we define the Chunk API shape as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required? |
| chunkId | String | Identifies the chunk | Yes |
| sizeInBytes | Integer | Number of bytes within the chunk payload | Yes |

Table : Type Definition for Chunk

When the job’s execution fails, the HTTP status reflects this as a 5XX error code. Otherwise, the result’s status code is 200.

## Describing a Single WAN Ingestion Job

The DescribeJob API takes a jobId as input and outputs a description of the job as follows:

| Name | Type | Description | Required? |
| --- | --- | --- | --- |
| sourceURI | String | Uniform resource indicator for the data source | Yes |
| chunkSizeInBytes | Integer | Maximum chunk size in bytes | Yes |
| numChunks | Integer | Total number of chunks in the job | No |
| description | String | Human-readable description of the job | No |
| creationDate | Date | When the job was created | Yes |
| timeToLiveInSeconds | Integer | Number of seconds after which this job is eligible for garbage collection | No |
| status | Enum | Most recent job execution status: {SUCCESS, FAILURE} | No |
| replicatedChunks | List [Chunk] | Successfully replicated chunks | No |

Table : Response Parameters for DescribeJob

When a job with the specified id does not exist, the HTTP status code in the result is 404. Otherwise, the status code is 200.

## Listing WAN Ingestion Jobs

ListJobs allows a user to page through all jobs that have been created and/or executed in the system.

It takes a single optional input parameter: pageToken. When specified, pageToken is used to provide the next page of results. Only recent page tokens returned in previous queries are considered valid.

The output of ListJobs consists of the parameters below. This API should always return an HTTP status code of 200 unless the page token is invalid (code 400).

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required? |
| jobs | List [ListedJob] | WAN ingestion jobs | Yes |
| pageToken | String | Token for next page of results | No |

Table : Response Parameters for ListJobs

We define the ListedJob API shape as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Description | Required? | |
| jobId | String | Identifies the WAN ingestion job | | Yes |
| executedSuccessfully | Boolean | True if and only if the job has been executed successfully | | Yes |

Table : Type Definition for ListedJob

## Getting Data for a Chunk

To fetch the data within a chunk, provide the chunkId to GetChunk. The output of GetChunk is defined below. When the specified chunk does not exist, the HTTP status code returned is 404. Otherwise, the status code shall be 200.

| Name | Type | Description | Required? |
| --- | --- | --- | --- |
| chunkDataAsBase64 | String | Payload of the chunk, encoded with Base64 | Yes |
| chunkSizeInBytes | Integer | Number of bytes within this chunk | Yes |

Table : Response Parameters for GetChunk

# LAN Ingestion Service

The *Local Area Network (LAN) Ingestion Service* replicates data chunks gleaned from the WAN Ingestion Service to a series of nodes within a local data center, rack, or other co-geolocated deployment. This ingestion phase proceeds in a series of successive execution “waves.”

Like deployments in a continuous delivery environment, users can define a threshold at which a replication wave is considered successful. Failed executions can be retried by re-executing the failed job.



## Creating a New LAN Ingestion Job

We define the CreateJob API to allow a user to create a new ingestion request. Its input parameters are as follows:

| Name | Type | Description | Required? | |
| --- | --- | --- | --- | --- |
| parentJobId | String | Associated WAN ingestion job | | Yes |
| batchSize | Integer | Maximum number of hosts in each replication wave | | Yes |
| successThreshold | Float | Fraction of hosts whose replication must succeed in each wave (a number between 0 and 1) | | Yes |
| description | String | Human-readable description of the job | | No |

Table : Request Parameters for CreateJob

In the event of successful request creation (HTTP status code 201), the API returns an identifier for the job (jobId) that was created. This API may return a status code of 400 when input validation fails.

## Executing a LAN Ingestion Job

ExecuteJob plans a series of successive ingestion “waves” consistent with the job parameters then executes each wave serially. Its only input parameter is jobId, which is required.

The output parameters for ExecuteJob are as follows:

| Name | Type | Description | Required? |
| --- | --- | --- | --- |
| status | Enum | Job execution status: {SUCCESS, FAILURE} | Yes |
| numFailedBatches | Integer | Number of batches that failed to execute successfully | No |

Table : Response Parameters for ExecuteJob

When the job’s execution fails, the HTTP status reflects this as a 5XX error code. Otherwise, the result’s status code is 200.

## Describing a Single LAN Ingestion Job

The DescribeJob API takes a jobId as input and outputs a description of the job as follows:

| Name | Type | Description | Required? |
| --- | --- | --- | --- |
| jobId | String | Identifies this job | Yes |
| parentJobId | String | Associated WAN ingestion job | Yes |
| batchSize | Integer | Max. number of hosts in each replication wave | Yes |
| successThreshold | Float | Min. percentage of hosts whose replication must succeed in each wave | Yes |
| creationDate | Date | When the job was created | Yes |
| numExecutions | Integer | Number of attempted job executions | Yes |
| status | Enum | Most recent execution status: {SUCCESS, FAILURE} | No |

Table : Response Parameters for DescribeJob

When a job with the specified id does not exist, the HTTP status code in the result is 404. Otherwise, the status code is 200.

## Listing LAN Ingestion Jobs

ListJobs allows a user to page through all jobs that have been created and/or executed in the system.

It takes a single optional input parameter: pageToken. When specified, pageToken is used to provide the next page of results. Only recent page tokens returned in previous queries are considered valid.

The output of ListJobs consists of the parameters below. This API should always return an HTTP status code of 200 unless the page token is invalid (400).

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required? |
| jobs | List [ListedJob] | LAN ingestion jobs | Yes |
| pageToken | String | Token to fetch next page of results | No |

Table : Response Parameters for ListJobs

We define the ListedJob API shape as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required? |
| jobId | String | Identifies the LAN ingestion job | Yes |
| executedSuccessfully | Boolean | True if and only if the job has been executed successfully. | Yes |

Table : Type Definition for ListedJob

# Use Case: 1 TB, Two Data Centers

The primary use case whose requirements drove the initial API for these services involves two data centers with 50 thousand nodes in each. To facilitate this use case, we would deploy both the LAN Ingestion Service and the WAN Ingestion Service in each data center. There are at least two different ways to proceed from here.

One choice we could make is to create two WAN ingestion jobs and execute them concurrently in each data center. Alternatively, we could wait for one WAN ingestion job to finish in the first data center, then use DC1 as the source for a second WAN ingestion to the other data center. Which choice we make will depend primarily on the networking characteristics between data centers versus the external data source. Either way, we can then execute LAN ingestion jobs at both sites to complete replication.

This design strives to generalize. It is our goal to support data ingestion processes with an arbitrary number of data centers and/or data sources. We also hope to provide flexibility for fluctuating network conditions and future requirements.