CONFIDENTIAL
2024-11-20

Sizing Guide for Cloud

© 2026 SAP SE or an SAP affiliate company. All rights reserved.

Content

1 Introduction. 3

1.1 Factors That Influence the Sizing. 5

2 SAP Data Intelligence Cloud Instances. 6

3 Sizing Calculation for SAP Data Intelligence Cloud. 7

3.1 Sizing of Applications. 8

3.2 Sizing for Data Governance. 10

3.3 Sizing for Data Pipelines. 11

Characteristics of Data Pipelines That Influence Sizing. 12

Simplified Sizing Procedure. 14

Custom Sizing for Data Pipelines. 14

3.4 Sizing of Replications. 18

1 Introduction

The Sizing Guide for SAP Data Intelligence Cloud explains the sizing for the diagnostics, cluster applications, tenant applications, user applications, and pipeline components of SAP Data Intelligence Cloud.

SAP Data Intelligence Cloud is a comprehensive data management solution that connects, discovers, and enriches disjointed data assets and then transforms them into actionable business insights at enterprise scale.

SAP Data Intelligence Cloud enables the creation of data warehouses from diverse enterprise data, simplifies the management of IoT (Internet of Things) data streams, and facilitates scalable machine learning. SAP Data Intelligence allows you to leverage your business applications to become an intelligent enterprise and provides a holistic, unified way to manage, integrate, and process all of your enterprise data.

Functions of SAP Data Intelligence Cloud

The capabilities of SAP Data Intelligence cloud can be partitioned into the following functional areas:

- Data Governance:
  Discover information by indexing, publishing, and profiling your datasets. Extract metadata and lineage to learn about the source and transformations of the dataset. Organize and label your datasets to help you find data quickly. Use data preparation to find data quality issues, correct and standardize data, and then output the data for analysis.
- Data Pipelines:
  The SAP Data Intelligence Modeler helps create data processing pipelines (graphs) and provides a runtime and design-time environment for data-driven scenarios. The Modeler reuses existing coding and libraries to orchestrate data processing in distributed landscapes.
- Replications:
  Replicate datasets from a source system to a target system.

SAP Data Intelligence Cloud Components

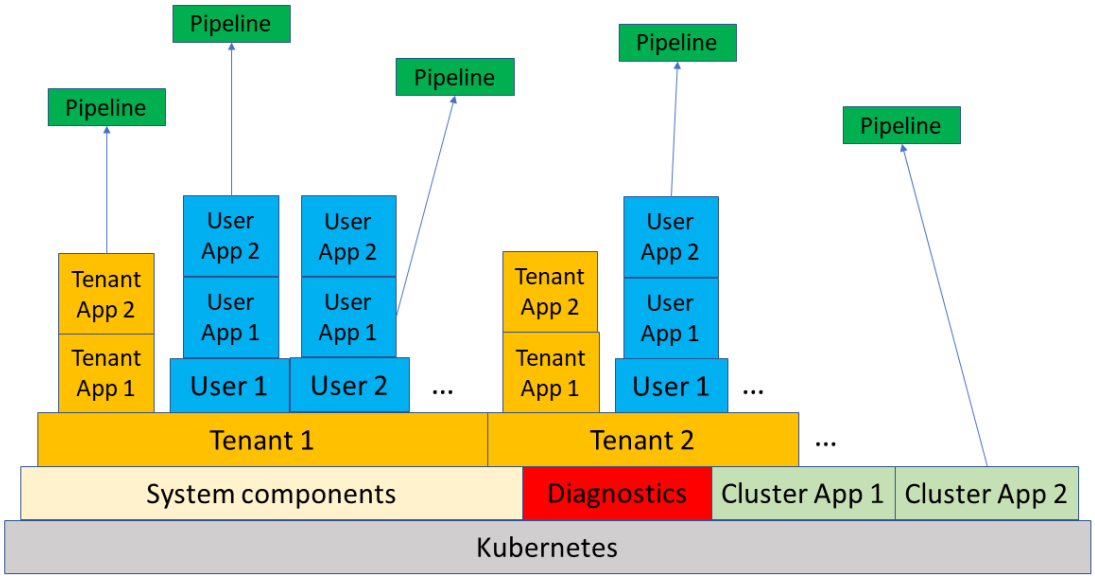
Multiple components comprise SAP Data Intelligence Cloud and factor into sizing.

The following table lists the system components that constitute the system management and central persistencies of SAP Data Intelligence Cloud, including an SAP HANA database, a storage gateway (to abstract connectivity to storages) and a user account and authentication server (UAA).

System Component	Description
Diagnostics	Services for monitoring of performance metrics and log messages.

System Component	Description
Cluster applications	Applications running one instance per cluster. This includes the Central Connection Management (CCM) and the Replication Management Service (RMS).
Tenant applications	SAP Data Intelligence Applications running one instance per tenant. This includes, for example, the Launchpad and the Metadata Explorer.
User applications	SAP Data Intelligence Applications running one instance per user. This includes the Pipeline Modeler.
Pipelines	All workloads, including those under the category of Data Governance, are executed in terms of pipelines. Pipelines run in separate Kubernetes pods (at least one pod per pipeline).

The following diagram shows the structure of SAP Data Intelligence:



Related Information

Factors That Influence the Sizing [page 5]

1.1 Factors That Influence the Sizing

The cloud resource requirements for SAP Data Intelligence Cloud in terms of physical memory (RAM) and CPU processing power can depend on a number of factors.

- The number of tenants.
- The total number of users across all tenants. This refers to the number of existing users or the number of simultaneously active users. See Custom Sizing for Data Pipelines [page 14].
- The applications used by these users.
- The number of simultaneously executed workloads (pipelines).
- The characteristics of these workloads (pipelines). See Custom Sizing for Data Pipelines [page 14].

The memory consumption of the system components, diagnostics, and applications constitutes static overhead, on top of which the highly variable workloads will execute. This static overhead is mostly determined by the first three factors listed above (tenants, users, applications) (see Sizing of Applications [page 8]). The workloads are a more complex topic (see Sizing for Data Governance [page 10] and Sizing for Data Pipelines [page 11]).

2 SAP Data Intelligence Cloud Instances

To understand the sizing considerations, you need to understand the structure and configuration of an instance of SAP Data Intelligence Cloud. An SAP Data Intelligence cloud instance runs on a Kubernetes cluster in the cloud infrastructure of one of the supported hyperscalers (typically Amazon Web Service (AWS) or Azure).

Initially, this Kubernetes cluster contains three Kubernetes worker nodes:

- Two for system node pool hosting the system components of SAP Data Intelligence Cloud.
- One for the worker node pool, which is subsequently filled by the applications and workloads started by users. (for example, the SAP Data Intelligence Cloud worker nodes are a subset of the Kubernetes worker nodes).

Each node of this cluster has a capacity of 8 virtual CPUs and 32 GB of RAM, whether it belongs to the system or worker node pool. While the system node pool remains of constant size (2 nodes), the worker node pool may grow based on an autoscaling feature: more nodes will be created if the applications and workloads started by users require more resources. Nodes will be removed when workloads are completed and the resources are no longer needed.

This defines the building blocks and the variables for the upcoming sizing considerations.

3 Sizing Calculation for SAP Data Intelligence Cloud

Estimate how much (for example, to how many nodes) the worker node pool will grow, given the characteristics of a specific customer use case.

The system node pool nodes are fixed. The worker node pool nodes are added based on the resource needs of applications and workloads, such as pipelines. For size calculation, you must estimate how many nodes the worker node pool will grow to given the characteristics of a specific customer use case.

Derive this number in terms of the following:

1. Static overhead based on the number of tenants, users, and active applications.
2. The total resource usage of concurrent workloads (pipelines) at peak times.

The result of the size calculation is an estimated amount of required CPU and RAM resources, and the expected number, “N”, of worker nodes is the smallest number so that both of the following conditions are fulfilled:

- `N * 8 vCPUs > total required CPU for static overhead and workloads`
- `N * 32 GB > total required RAM for static overhead and workloads`

Including the system node pool, the total number of cluster nodes will be N+2.

Regarding static overhead, you can restrict yourself to cluster, tenant, and user applications. Other components, for example the overhead imposed by Kubernetes itself, or the pods responsible for log and performance metrics collection, consume considerably less resources, usually less than 5% of a node's capacity, which is certainly smaller than the uncertainty of our estimates.

The static overhead of the applications refers to the memory and CPU consumption when they are idle, (when no workloads are running that use these applications). In the case of CPU this will in general be the associated CPU request. Even if an application does not use the CPU at all for a certain time period, from Kubernetes perspective the defined CPU request is always reserved and therefore counts as static overhead. In the case of memory, the static overhead is the maximum of the memory request and the actual memory usage.

The workload characteristics connect with the functions of SAP Data Intelligence and how these functions are used.

The main questions in regard to the estimation are:

- How many tenants are expected on the DI Cloud instance (excluding "system" and "support")?
- How many users (Administrators, Developers, Data Governance) are expected across all tenants?
- How many concurrent workloads are expected in the area of Data Governance (data profiling / publishing / preparation) at peak times?
- How many concurrent data pipelines are expected at peak times?
- What are the characteristics of those pipelines? I.e., which operators are used? Are there throughput requirements which are to be satisfied by scaling out a pipeline ("group multiplicity")?
- How many concurrent replications are expected at peak times?

Data Governance and Replication workloads use certain standard pipelines, for which the resource requirements are easier to estimate. For more information, see Sizing for Data Governance [page 10] and Sizing of Replications [page 18]. For generic data pipelines, the situation is more complicated, since there is a vast number of possibilities. In Sizing for Data Pipelines you can see the general considerations that allow for a rough estimate, but then also study in detail the sizing procedure for a pipeline with given throughput requirements.

Following the structure of this topic, the total cloud resource requirements for the DI worker nodes can be divided as follows:

- RAM

```
RAM_TOTAL = RAM_STAT (Static RAM required for Applications)
          + RAM_GOV (RAM for Data Governance workloads)
          + RAM_PIP (RAM for Data Pipelines)
          + RAM_RMS (RAM for Replications)
```

- CPU

```
CPU_TOTAL = CPU_STAT (Static CPU required for Applications)
          + CPU_GOV (CPU for Data Governance workloads)
          + CPU_PIP (CPU for Data Pipelines)
          + CPU_RMS (CPU for Replications)
```

An example for applying the formulas above is shown in Example Sizing for a Reference Scenario [page 16].

Due to architectural changes in the software and because of additional or new findings (from customer implementation projects as well as from observations and/or tests in SAP-internal systems) formulas can change and will be adapted. Therefore it is advisable to regularly check this document for updates and to always refer to the newest document version.

Related Information

Sizing of Applications [page 8]
Sizing for Data Governance [page 10]
Sizing for Data Pipelines [page 11]
Sizing of Replications [page 18]

3.1 Sizing of Applications

Learn more about the cloud resource requirements which constitute the static overhead on the nodes of the worker node pool, for example, the resource requirements which are independent of the actual workloads.

You can get a reasonable approximation for this when you restrict yourself to the SAP Data Intelligence applications and consider the following:

- The CPU requests of the application pods.
- The maximum of the memory requests and the actual (idle) memory usage of the application pods.

For each application, check the default request settings and measure typical memory consumption, which then yields the following formulas (assuming that all major applications are running):

- RAM

```
RAM_STAT = Number of tenants (besides the system tenant) * 5 GB
         + Number of users (across all tenants) * 0.2 GB
         + 1 GB
```

- CPU (in units of vCPUs)

```
CPU_STAT = Number of tenants (besides the system tenant) * 1
         + Number of users (across all tenants) * 0.1
         + 1.5
```

In the case of a system with many users (for example, 100 users), User Applications contribute a significant part to the system's resource consumption. For such a setup, one should consider activating the TTL (time-to-live) feature for applications, which hibernates User Applications for users who have been inactive for a given time.

With this configuration, only the number of currently active users contributes to the formulas. Without this configuration, the number of existing users is what matters. With activated TTL, there will be some delay (typically around 30 seconds) when a user logs in again, before the applications are available, because they need to be restarted.

The following applications have been taken into account (in alphabetical order):

Application	Type	Remark
Axino	Tenant	Pod axino-service-*
Connection Service	Cluster	Pod connection-service*
Core Application	Tenant	Pod datahub-app-core-*
Data App Daemon	Tenant	Pod datahub-app-daemon-*
Flowagent	Tenant	Pod data-hub-flow-agent-*
Launchpad	Tenant	Pod datahub-app-launchpad-*
Metadata Explorer	Tenant	Pod datahub-app-data-*
Modeler UI	Tenant	Pod modeler-ui-*
Pipeline Modeler	User	Pod pipeline-modeler-*
Replication Management	Cluster	Pod rms-*
Scheduler	Tenant	Requests are forwarded to the core application.
Shared	Tenant	Pod shared-*
System Management	Tenant	Pod datahub-app-system-man-*

3.2 Sizing for Data Governance

SAP Data Intelligence helps you to govern data across different systems by using the Metadata Explorer, which can gather information about the location, attributes, quality, and sensitivity of data.

Data Governance jobs are executed in terms of certain standard pipelines. They can be divided into four different types:

- Profiling
- Publication
- Preparation
- Rules

The pipelines for data governance are single-group pipelines, meaning each pipeline runs in a single pod. A job refers to a certain data set (to be profiled, published, prepared, checked for rules) stored in some system external to SAP Data Intelligence. (Publication jobs can refer to multiple data sets.)

The main questions in regard to Sizing for Data Governance are:

- How many concurrent jobs are expected for data governance at peak times?
- How many of those belong to each type?
- What kinds of data sets do they refer to?

The number of concurrent data governance jobs influences cloud resource requirements because each job consumes resources. The more jobs running at a given point in time, the more memory and CPUs required. If you run jobs sequentially, the requirements per job do not add up, but the system needs to have enough memory and CPUs for the most demanding job.

In addition to the type of the job, a number of other factors influence the memory consumption, in particular:

- Type of the data set (file or table).
- Data source (where the data set is stored).
- Size of the data set in terms of rows.
- Size of the data set in terms of columns.
- Data types/data distribution inside the data set.
- File type (.csv, .orc, .parquet).
- For preparation jobs: the number of transformations and size of the target data set.

The CPU usage of data governance is typically between 1 and 2 vCPUs.

To formulate an estimate for the resource requirements, absorb the specifics of memory usage into a factor "AVG_TYPE_FACTOR", to be specified further below. The simplified formulas to derive the resource requirements are as follows:

- RAM

  ```
  RAM_GOV = Number of concurrent jobs * 2 GB * AVG_TYPE_FACTOR
  ```

- CPU (in units of vCPUs)

  ```
  CPU_GOV = Number of profiling jobs * 2
  ```

Here, AVG_TYPE_FACTOR is the estimated average of TYPE_FACTOR across all concurrent jobs, and TYPE_FACTOR is defined as (Memory usage of a job / 2 GB). The reasoning behind this is that the memory

usage in many cases will be around 2 GB, and therefore TYPE_FACTOR close to 1. Its exact value depends on the type of job and on some specifics of the data, or, in the case of preparation jobs, on the type of transformations:

- Profiling Jobs: Profiling data allows to determine the statistics of a data set, like the number of null or blank values, distinct and unique values, minimum and maximum and average length values. TYPE_FACTOR should be taken as 1 if the data set is a database table, 2 for any other types (for example, the data set is a file in cloud storage).
- Publication Jobs: Publication makes data sets available in the metadata catalog. A TYPE_FACTOR of 1 can be assumed.
- Preparation Jobs: Preparation jobs allow you to transform data sets. These jobs may demand more memory than other types, depending on the type of transformation. TYPE_FACTOR should be set to 2 in general, but up to 5 in case of JOIN/UNION/AGGREGATE operations.
- Rules Jobs: Rules help to determine whether the data complies with business constraints and requirements. The TYPE_FACTOR is 1 in general, but for very wide tables/files (more than 250 columns/ fields), the job's memory limit should be increased to 4 GB, corresponding to a TYPE_FACTOR of 2.

3.3 Sizing for Data Pipelines

Sizing of Data Pipelines support a flow-based programming paradigm for data ingestion and transformation.

The main questions to answer for Data Pipelines are:

- How many concurrent data pipelines are expected at peak times?
- What are the characteristics of these pipelines?

The cloud resource requirements for data pipelines are determined as the sum of the individual requirements for each of the concurrent pipelines at peak times. The number of concurrent data pipelines influences resource requirements because each pipeline consumes resources. Simply spoken, the more pipelines are running at a given point in time, the more memory and CPUs are required. If pipelines are executed sequentially, the requirements per pipeline do not add up, but the system needs to have enough memory and CPUs for the most "demanding" one.

The range of possibilities for data pipelines is very large, and resource requirements may differ a lot. It is strongly advisable to measure the memory and CPUs required for your pipelines early and incorporate the learnings into sizing of your (production) environment.

Therefore, this section is structured as follows:

1. Discuss the characteristics of data pipelines that influence the sizing.
2. Define conditions under which a simplified sizing calculation is sufficient, in terms of a rough estimate, and provide the formula of this estimate.
3. For the more complex cases, you can learn the procedure for how to estimate the resource requirements of a more demanding pipeline. This procedure requires that a test environment for SAP Data Intelligence Cloud is already available.

> **Note**
>
> A data pipeline may create a significant impact on "connected" systems. For example, the source or target systems the pipelines read from or write to. This sizing guide does not advise how to size "connected" systems.

Related Information

Characteristics of Data Pipelines That Influence Sizing [page 12]
Simplified Sizing Procedure [page 14]
Custom Sizing for Data Pipelines [page 14]

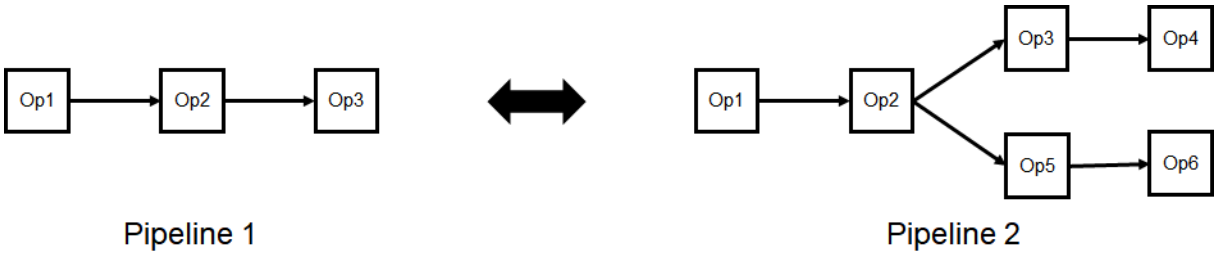
3.3.1 Characteristics of Data Pipelines That Influence Sizing

The resource requirements for data pipelines are influenced by a number of factors, such as pipeline complexity and amount of data to process.

The following sections discuss the various factors that affect size requirements for pipelines.

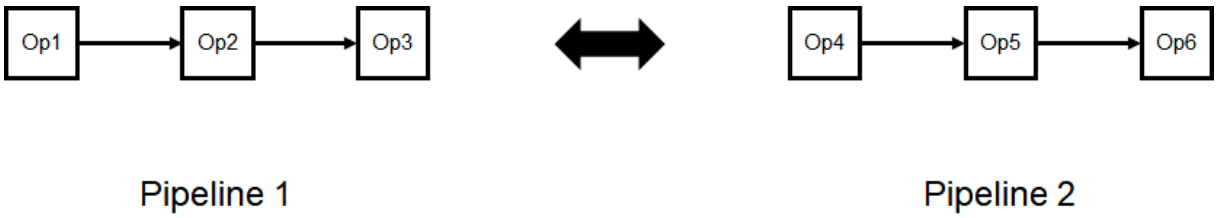
Data Pipeline Size

The size of a pipeline can influence memory and CPU requirements. Bigger pipelines need more memory and CPU compared to smaller pipelines. The following pipeline diagrams compare a small pipeline to a bigger pipeline.



Operators

CPU and memory requirements vary between operators. Therefore, pipelines of the same size can have different needs based on the operators in the pipeline.



Complexity of Logic

Two operators that are the same can be configured to perform simple or complex processes. Particular processing operators, such as JavaScript, Go, Python, or R allow you to develop either simple or complex logic.

Groups and Multiplicity

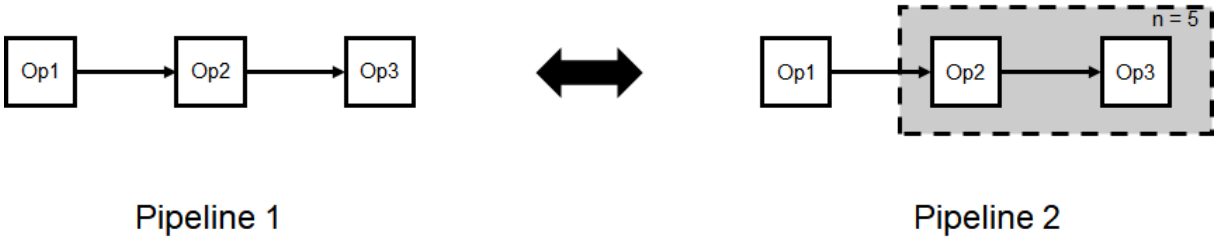
When you define groups in a pipeline and assign multiplicity to these groups, the pipeline spawns multiple pods. Multiplicity can increase memory and CPU requirements when compared to the same pipeline that doesn't use groups.

Group multiplicity increases the throughput of a pipeline. Multiplicity parallelizes a workload by creating multiple copies of a set of operators, where each copy processes a certain part of the data.

In the following diagrams, only one pod is spawned from pipeline 1, but seven pods are spawned from pipeline 2:

- One pod for operator 1;
- Five pods for the group that consists of operator 2 and 3;
- One pod, `vflow-bus` (not shown in the picture), that communicates between the groups.

A data pipeline using multiplicity N for all operators has roughly N times the resource requirements compared to the same data pipeline not using multiplicity.



Processed Data Volume

The data volume processed by a pipeline can affect resource requirements.

> **Example**
>
> A pipeline receives a small amount of data. Consequently, the operators are idle most of the time. However, if the same pipeline receives a large amount of data to process, then the operators are less idle and consume more memory and CPU.

3.3.2 Simplified Sizing Procedure

A simplified sizing calculation for pipelines may be possible in certain situations.

The simplified sizing could be considered if all of the following conditions are met:

- The expected number of concurrent pipelines at peak times is relatively low, usually smaller than 20.
- Each pipeline contains only one group and does not use multiplicity.
- All operators process data in batches not bigger than 20 MB.

In this case, the resource requirements of pipelines will be smaller than those of the static components (implying that the error of our estimate will be less significant), and typically 1 GB of memory and 1 vCPU will be sufficient for each pipeline.

The simplified formulas to derive the resource requirements are then:

- RAM

```
RAM_PIP (in MB) = Number of pipelines * 1 GB
```

- CPU (in units of vCPUs)

```
CPU_PIP = Number of pipelines
```

3.3.3 Custom Sizing for Data Pipelines

The custom sizing of a data pipeline is done using a set procedure.

In the context of this procedure, the pipeline contains single input and output operators, which interface to an external data source and consumption target, respectively. Between the input and output operators, the transformation is performed by a custom operator or a sequence of custom operators. Input, transformation and output operators are grouped for multiple instance parallel execution (group multiplicity), see example in the image below (Fig. 1).

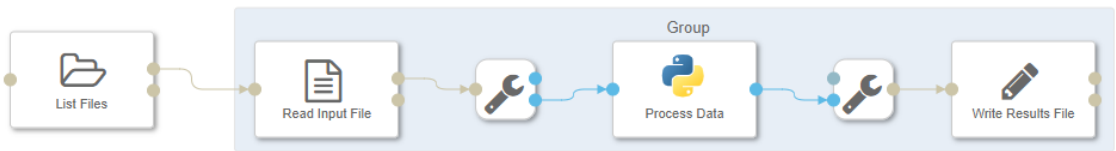


Figure 1: Example pipeline comprised of data input, output, and transformation operators. Operators are combined in a Group in preparation for parallel execution. The operator outside the group provides information on the workload to the pipeline instances.

Typically, the transformation operator will be a "Process Data" operator with logic coded in one of the available programming languages (Python, Go, Javascript, R). Our example pipeline uses Python. The decision on which language is used depends on coding efficiency choices and transformation complexity. For very high throughput requirements on a numerically complex transformation algorithm, a Go-implementation might be considered.

Input to the sizing procedure is the target throughput, typically given as data volume or number of records which are to be consumed by the pipeline, per second. Output will be resource requirements, specified in number of CPUs and amount of memory.

Related Information

Sizing Procedure [page 15]
Example Sizing for a Reference Scenario [page 16]

3.3.3.1 Sizing Procedure

The sizing procedure can be described in five steps. In Example Sizing for a Reference Scenario, the procedure by application to a reference scenario will be illustrated.

1. Create test environment.
   Set up the transformation pipeline in a production-like test environment.
2. Create boundary conditions.
   Establish the input/output boundary conditions. The external capabilities must be checked to make sure the target throughput can be reached at all in the given environment. This can be achieved by checking the external SLAs for throughput bandwidth, latency, and parallel access capability. If no external information is available, these quantities can be established by measurements using an appropriate no-operation pipeline (Fig. 2).

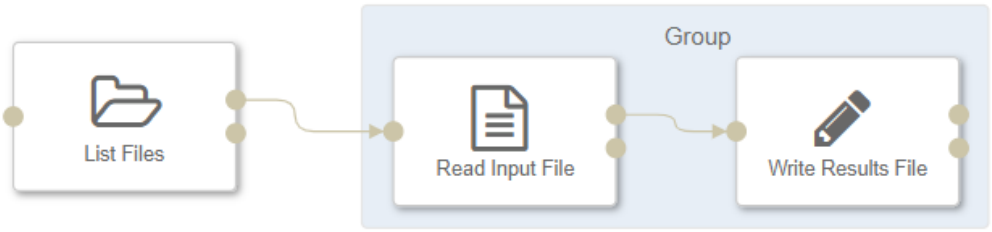


Figure 2: No-operation pipeline related to the example transformation pipeline shown in Fig. 1.

3. Run single instance measurement.
   Run a single instance of the transformation pipeline (parallelity = 1) with stable throughput for a defined time. Measure the related CPU and memory consumption on the execution node, as well as the throughput

for this execution. CPU and memory usage can be measured using the Monitoring App. Throughput can, for example, be determined from the data volume recorded on the consumption target. From the measurement, extract the number of CPUs and memory required for execution, and the observed single instance throughput (CPU_Single, RAM_Single, THROUGHPUT_Single).

4. Calculate sizing.
   Calculate the required parallelism (PARALLELITY), number of CPUs (CPU_PIP), and amount of memory (RAM_PIP) required for the raw execution of the target throughput. Use the linear formula:

   ```
   PARALLELITY = Target_Throughput / THROUGHPUT_Single
   ```

   ```
   CPU_PIP = PARALLELITY * CPU_Single
   ```

   ```
   RAM_PIP = PARALLELITY * RAM_SINGLE
   ```

5. Finalize sizing result.
   Add the offset for the applications (CPU_STAT, RAM_STAT, see Custom Sizing for Data Pipelines [page 14]). The resulting sizing numbers for number of CPUs and amount for memory (CPU_SIZED, RAM_SIZED) serve as input for the configuration of the productive cluster.

   ```
   CPU_TOTAL = CPU_PIP + CPU_STAT
   ```

   ```
   RAM_TOTAL = RAM_PIP + RAM_STAT
   ```

   > **Note**
   >
   > - Depending on the choice of cluster configuration, the required parallelism for the target throughput might differ slightly (10-20%) from the calculated value PARALLELITY. Adjust the parallelism to the value needed in the productive environment.
   > - Load balancing in the Kubernetes cluster can be influenced by setting Group resource requests in the Group parameters. Set the CPU and memory requests to the values found in the single instance test measurement. The group resource requests can be varied for load balancing optimization.
   > - If multiple pipelines are executed concurrently, each of them contributes a term CPU_PIP and RAM_PIP in the formula above.

3.3.3.2 Example Sizing for a Reference Scenario

The example python transformation pipeline is shown in Fig. 1 in Custom Sizing for Data Pipelines.

The specific task performed by the pipeline is the transformation of data contained in comma-separated-value (csv) files, located on regional Google cloud storage (GCS). The transformed data are again written into csv files on GCS. The input data consist of records of an IoT-Showcase scenario. Each record contains a string, 2 integers, and 39 float values.

The transformation consists in the injection of interpolation records between the lines of the input data, containing the average of consecutive lines for the 39 float values. In the reference example, the input and output files contained 180k and 360k records or 46.3 and 92.5 MB in data volume, respectively. For the example sizing, a target throughput of 200k input records (50 MB) per second was set.

1. Test environment.
   The test was set up on a 5-node cluster on GCP, with 16 vCPUs and 64 GB per node. The cluster was connected to regional GCS storage. 720 individual input files of 46.3 MB each were located on the GCS input location.
2. Boundary conditions.
   The Google Cloud Storage SLAs at test time allowed one change per second per object. This does not limit the test as objects (output csv files) are individually written, once. There are explicitly no limits in the GCS SLA on writes across multiple objects or reads on objects. By running a no-operation pipeline (see Fig. 2 in Sizing Procedure [page 15]) with parallelity 40, both read- and write throughput capability at well above 1 GB/s was established.
   With this information, no boundary limitations due to service conditions or infrastructure are expected for the target throughput of 50 MB/s and 100 MB/s for read and write, respectively.
3. Measurement.
   The example pipeline with parallelity 1 was run for several minutes. During this time, no other activity was triggered on the cluster. The throughput was measured by counting the number of output files created during the runtime in the GCS output location. The result was 15 files, or 1.4 GB written in 420 s, converting to a write throughput of 3.3 MB/s or 13k Records/s, and a read throughput of 1.7 MB/s or 6.5k Records/s. The CPU and memory consumption were determined using Grafana dashboards on the information provided by the DI diagnostic server. On the cluster node executing the main pipeline pod, an increase was seen of the number of CPUs utilized by 1.0, and the memory volume utilized by 2.2 GB, during the execution of the test.
4. Raw sizing calculation.
   With the values from step 3 for single instance throughput, CPU and memory consumption, the following raw sizing results for the target throughput of 200k Records/s (all results rounded up to the next integer) was retrieved:

   ```
   PARALLELITY = Target_Throughput / THROUGHPUT_Single= 200k Records/s / 6.5k Records/s = 31
   ```

   ```
   CPU = PARALLELITY * CPU_Single = 31 * 1 = 31
   ```

   ```
   RAM = PARALLELITY * RAM_SINGLE = 31 * 2.2 GB = 69 GB
   ```

5. Finalize sizing result.
   With the results from Step 4, and the values for static components of the cluster (assuming just 1 tenant with 1 user, see Custom Sizing for Data Pipelines [page 14]), the following resource requirements were retrieved:

   ```
   CPU_TOTAL = CPU_PIP + CPU_STAT = 31 + 2.6 = 33.6
   ```

   ```
   RAM_TOTAL = RAM_PIP + RAM_STAT = 69 GB + 6.2 GB = 75.2 GB
   ```

   As a consequence, during the runtime of this pipeline, the "worker" node pool will grow to 5 nodes (5x8 vCPUs = 40 vCPUs, 5x32 GB RAM = 160 GB RAM). Including the nodes of the "system" node pool, the DI Cloud instance will consist of 7 nodes.

3.4 Sizing of Replications

Replication Management Service (RMS) allows you to replicate a number of tables (or other data structures) from a source system to a target system.

Sizing of Replication Flows in Data Intelligence Cloud

Like all other workloads, replications are executed in the form of pipelines, which are called worker graphs in this case. The worker graphs run in the background when replication flows are started via the replication flow user interface. Worker graphs are internal and cannot be modified like regular pipelines being created by a user in the Modeler application. A replication flow allows one tenant to replicate an arbitrary number (N) of data sets (for example, tables or CDS views) from one source system to one target system. Each table constitutes a task of the replication flow. All data are replicated in relatively small chunks (typically between 10 MB and 30 MB during the initial load), such that the memory usage of the worker graphs remains constant.

Each worker graph has five connections to the source and target system; that means five parallel threads can read and write data. The total number of connections (threads) used by a replication flow is a configurable parameter. This can be configured in the Monitoring application, and the default value is 10. In this case, two worker graphs are started for each replication flow. The general formula where the result has to be rounded up to the next integer is as follows:

```
Number of worker graphs per replication flow = Number of connections per
replication flow / 5
```

To scale out replications in order to achieve better throughput, there are two options:

Option	Result
Increase the number of replication flows. For example, don't replicate all tables as tasks within one replication flow, but distribute the tables (tasks) over several replication flows that can run in parallel.	Increasing the number of replication flows will create for each replication flow background processes in the source and target systems. Scale them carefully so as not to overload the source or target system with background processes.
Increase the number of connections within a replication flow to get more threads and, equivalently, more worker graphs.	With the start of multiple connections to the source and target systems, background processes get blocked for the handling of the data packages. Scale the number of connections carefully to not block too many resources in the system.

> **Note**
>
> For increasing the number of replication flows, you do not need to specify one replication flow table, but you can find a mix based on your use case to group the tables to achieve your desired performance.

Scale-out will work only up to a certain point, because at some point a threshold in throughput will be reached due to constraints of the source and target systems, or due to other limitations. Therefore, up to this threshold (which depends on the details of the connected systems), the trade-off between higher throughput and higher resource consumption (higher TCO) versus lower throughput and lower resource consumption (lower TCO) is a matter of configuration.

The resource consumption of a worker graph is relatively predictable. It usually consumes around 2 GB of RAM and between 1 and 2 vCPUs. If we assume roughly 1.5 vCPUs on average, the RMS formula for RAM and CPU becomes (with A the number of replication flows running in parallel and B the configured number of connections per replication flow) as shown below:

```
RAM_RMS = A * [B/5] * 2 GB
```

```
CPU_RMS = A * [B/5] * 1.5
```

If there is a certain requirement for total throughput (in terms of [MB/s] or [GB/h]), and if this required throughput is still in the realm of linear scalability for the given constellation, it can be achieved by configuring A and B such that the throughput that is achieved per worker graph depends on the specific scenario. However, from various measurements it is known that around 5 MB/s (uncompressed data volume) is often a good estimate.

```
A * [B/5] >= (required throughput) / (throughput per worker graph)
```

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

© 2026 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see https://www.sap.com/about/legal/trademark.html for additional trademark information and notices.