



The Retail Innovators

SAP Dynamic Pricing by GK

Sizing Guide

Version: v4.0.0



COPYRIGHT

© 2022 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.

1. You may not use the SAP Material for a purpose competitive with SAP or its products unless otherwise clearly permitted by applicable law.
2. You may not use the SAP corporate logo.
3. No use of other SAP trademarks is granted under this section. For information regarding use of SAP trademarks, see <http://www.sap.com/corporate/en/legal/trademark.html>.

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.

Internal Document Information: 1223593652 | 2022-03-02

TABLE OF CONTENTS

1	Introduction	4
2	How to size your Service	4
2.1	Involved Services	4
2.2	Which Resources are demanded	4
2.3	Sizing Process	4
2.4	Cloud Resource Limits	5
3	Key Drivers for the Resource Demand of the Dynamic Pricing Service	6
3.1	Understand how KPIs impact the Sizing	6
3.2	Most important Resources for each Process	7
3.3	Key Drivers for Sizing the Application Service	8
3.4	How Tenants influence the Sizing of the Database Service	9
3.4.1	Database Connections	9
4	Scaling Options	9
4.1	Scaling Options for a Single Node	9
4.1.1	Controlling Concurrency of Jobs	9
4.2	Horizontal Scaling with more Nodes	10
5	Performance Figures of the Reference System	11
5.1	Reference System Hard- and Software	11
5.1.1	Hardware and Software.....	11
5.2	Example for Resource Consumption on the Reference System	12

1 Introduction

This document describes performance figures and provides guidance for sizing of the application for various scenarios.

We assume that you are already familiar with the following documents to understand all architectural options:

- Architecture Guide
- Integration Guide

2 How to size your Service

2.1 Involved Services

The dynamic pricing service is made up of two services:

- The dynamic pricing application container: including OS, Java Runtime, Tomcat, AIR platform and application binaries
- The database service: default is to link a managed cloud database service. An other option is to setup a database container.

2.2 Which Resources are demanded

These are the resources needed to run the service:

Application Service	Comment
CPU	
RAM	
Internal Container Disk	Usually quota is 2 GB at no extra cost
External Storage	Should be a fast disk for temporary price computation data. No backup needed, does not have to survive a container restart.
Bandwidth	
Loadbalancer	For HTTPs, not for load balancing requests Depends on cloud if this is part of the offer or needs extra spending
VPN Gateway	Depends on the cloud platform if this is needed for communication with the on premise backbone systems.

Database Service	Comment
CPU	
RAM	
Storage	To persist the data.
Connections	
Backup Storage	Depends on how much data should be backed up if this is a managed cloud database service.

For scaling the application service you can add more pricing application containers (so called: compute nodes). This is helpful as the cloud container runtimes are limited in vertical scaling of CPU/RAM and so you need to scale out horizontally using compute nodes.

2.3 Sizing Process

Let's outline the sizing process:

1. Define your project's business KPI values: We provide you a template with key drivers for the resource demand. Fill them with your concrete demand

2. Based on the KPIs you can calculate the resource demand of your service. You can do that manually based on the guidance in this guide or you can use the "Dynamic Pricing Sizing Calculator" spreadsheet.
3. Having all resources at hand you need to map that to your concrete cloud environment:
 - a. Take into account the limits of the cloud environment, e.g. max. CPUs per container, etc.
 - b. Optionally: Take into account the classification of the resources, e.g. there is a given set of database categories with fixed resources (e.g. CPU, RAM, storage). So you need to pick the right category for your needs.
 - c. Involve more application containers to scale out your service if you exceed container runtime limits for one node

2.4 Cloud Resource Limits

We currently support those cloud platforms:

- Microsoft Azure Cloud

At the time of writing these limitations did apply to those platforms:

Resource	Microsoft Azure	Microsoft Azure (Kubernetes)
Container Runtime: max. RAM per node	14 GB	432 GB
Container Runtime: max. CPUs	4	64
Database: CPU	2 - 64	2 - 64
Database: RAM	No manual setting	No manual setting
Database: Storage	20 GB - 3999 GB	20 GB - 14 TB

Please don't forget to review the current limitations of you platform as these might have changed meanwhile.

3 Key Drivers for the Resource Demand of the Dynamic Pricing Service

3.1 Understand how KPIs impact the Sizing

KPI	Impact on Application Sizing	Impact on Database Sizing
Total number of products	CPU+, RAM+, Storage+	Storage+++, CPU+
Total number of categories		Storage+
Competitor Prices/product	RAM+	Storage++
Competitor Price Quota in %	CPU+	Storage++, CPU+
Competitor Prices every x days	CPU+	CPU+
Masterdata in Days	CPU+	Storage+++
Historical Transactions in Days	CPU++, RAM+++	Storage+++
Order Transactions/day	CPU+, RAM++, Storage+	Storage+++
Click Transactions/day	CPU+, RAM++, Storage+	Storage+++
Order Transactions: unique items	CPU+, RAM+++ , Storage+	Storage+++
Click Transactions: unique items	CPU+, RAM+++ , Storage+	Storage+++
Product changes per week		Storage+++
Number of price runs/day	CPU+++	CPU++
Number of price calculations/run	CPU++, Storage+	CPU+
Historical Statistics in Days		Storage+
Simulations/Day	CPU+++	CPU+, Storage+
Maximum minutes for price run	CPU+++	CPU++
Transactions in Days/Computation	CPU+++ , Storage++	CPU+, RAM+
Simultaneous Computations	CPU+++ , Storage++	CPU+, RAM+
Segments		
Strategies/Segment	CPU++	
Historical Simulations in Days		Storage+
Historical Price results in days		Storage+

3.2 Most important Resources for each Process

Process	Most significant resources
Importer	Application storage and database storage At first, all importers store the uploaded data into the file system. After the file is uploaded the processing starts. The files are extracted, validated, reorganized and finally imported into the database. After the import, the temporary files in file system should be removed, so the space is free again after that.
Item Import	Database storage Every master data update inserts a new entry in database and leaves the old one as historic entry in the database. The old entries are removed by cleanups only. This means the more updates, the more the database grows until the cleanup limit is reached.
Competitor Prices import	Database storage Competitor prices are stored additionally to the products, but they could be very large because the number of entries is multiplied with the number of competitors. Competitor prices are versioned too, so every competitor price update, adds one more version of price entries into the database.
Transaction Log import	Database storage Transactions are stored in a aggregated entry, because only counts per day are stored. <u>The transactions are aggregated over the following fields:</u> day, product id, price, netto price, channel, all extension fields, transaction type
Price computation	Database storage Every price computation creates prices as result which are stored in the database. Additionally there is a execution protocol with information about the price computation runtime. Application storage The price calculation needs temporary disk space for storing some necessary files for the execution. Memory The price computation is a very memory consuming process. The memory consumption depends mostly on how many days of transactions are used for the computation, because all used days must be analyzed in memory. CPU For a price calculation one full CPU of the AIR server is used. The calculation task itself runs as single thread, but it is possible to run multiple price computations at the same time. Time The price computation could last a long time(hours) depending of the number of historical transactions and the number of products. The run time of price calculations should be considered for the scheduling of the computations and the scheduling of external systems which want to use the price computation results.
Segment Statistics	Database storage The execution of tasks in the AIR server always produce statistical data (KPI statistic, segment statistic, simulation statistic). Every statistic is stored in the database, There is a cleanup setting to limit the statistic history.
Simulation	Database storage Price simulations create a simulation result and the simulation statistic in the database. The stored historical simulations could be limited by setting. Application storage Simulations are repeated price computations over many days for two candidate strategies. So we have 2 * "number of days" computations to execute. The two strategies could run at the same time so we need twice as much temporary disk storage. Memory Because two calculations run at the same time we need twice as much memory too. CPU The two strategy candidates of the simulation are using full CPU core each. Time The time of the simulation in total is ("number of days" * "time of one computation") + "calculation simulation stastic".
Export	RAM, CPU Exports create file streams from pricing computation results as CSV or XML. Depending on the exporter those files are stored in file system or they are sent to a consumer system.

3.3 Key Drivers for Sizing the Application Service

Process	Drives	Comment
Import of Data	RAM+, Storage+, CPU++	Importers first download the data and store it temporarily on storage. Then they stream the data from disk. Generally there isn't a dependency on between amount data and RAM. However, some importers need lookup caches that are preloaded before the import and kept until the end of the import. This is true if you are running master data importers in update mode. In initial mode they also use lookup caches that grow with the number for data in the database - but just for existency check and thus RAM demand isn't that high.
Export of Data	RAM++, CPU++	For the export all optimized prices are loaded from database into memory and then exported. The export to the destination uses streaming.
Price Computation	RAM+++, CPU+++, Storage++	<ol style="list-style-type: none"> 1. The price computation saves all items and all aggregated transactions to disks before computation. So you have to make sure your disk storage is large enough. 2. The price computation loads all items to memory and also all aggregated transactions to memory. There is a strong relationship to the RAM demand of your application. The price strategy parameter defining the days back in history for the transactions is a crucial parameter that has to be carefully reviewed. The amount of aggregated transaction data depends on the days back in history and how well the aggregation feature works (see details below) 3. Prices are optimized: this is where the CPU comes into the game 4. Optimized prices are stored in database (see next section for resource impact) 5. Segment and price strategy statistic calculation is performed: Mostly CPU of database (see next section) 6. Cleanup of the disk storage
Simulation	RAM+++, CPU+++, Storage++	If a simulation compares historical statistical data with a newly configured price strategy, the resource consumption resembles the one of a single price computation. If a simulation compares two (or more) price strategies, then the price computation resources are at least doubled.

It's important for the sizing to estimate the peak resource usage.

Which processes can be distributed (and serialized) over 24 hours of a day? How many processes are running at the same time?

If you are running more simulations at the same time and also run the scheduled price computations at the same time, then your peak demand is very high.

Don't forget to take into account your business accounts - each of them can run imports, export data, etc.

Key Drivers for Sizing the Database Service

Process	Drives	Comment
Import of Data	Storage+/***/+++, CPU++	<p>CPU: Importers store data in batches. Most importers use a separate thread for sending data to the DB. If there is no load on the DB one importer can consume almost up to one DB core.</p> <p>Storage: This depends on the kind of import: Transactions occupy much space, exchange rates almost nothing. The amount of aggregated transaction data depends on the days back in history and how well the aggregation feature works (see details below)</p> <p>One item set needs not much storage. However, as item data is stored in historical versions, the storage demand depends a lot on how long you store the historical item data versions and how many items are changed per day. Thus it is recommended to use the update modes of the masterdata importers and feed them only with the new and updated items per day. This saves a lot of storage space.</p>
Export of Data	RAM+, CPU+	For the export all optimized prices are loaded from database into memory and then exported.
Price Computation	RAM+, CPU++, Storage+	<ol style="list-style-type: none"> 1. The price computation loads all items of the segment(s) and all aggregated transactions from DB. The price strategy parameter defining the days back in history for the transactions is a crucial parameter that has to be carefully reviewed. 2. Optimized prices are stored in database. This depends on the item counts that are optimized. 3. Segment and price strategy statistic calculation is performed mostly using the database. It depends on how many items have been optimized.
Sgement Statistics	RAM+, CPU+	Daily segment statistic calculation is performed mostly using the database. It depends on how many items have been optimized.
Simulation	RAM+, CPU+, Storage+	It's similar like price computation process. The price computations are executed in a serial way one after the other.

How the transaction aggregation works

Transactions are aggregated on a daily basis for each item id and for order/click transactions separately. If there is another order transaction for that given day with a different item price, then a new record is written.

Examples:

- 5x order tx for item 1 with price 8.99 for a day and 3 click transactions for item 1. Aggregation results in two records in the database (import) and RAM (for computation):
Order, 1, 8.99, Date
Click, 1, Date
- 5x order tx for item 1 with price 8.99 and 1x for item 1 with price 7.99. Aggregation results in two records in the database (import) and RAM (for computation):
Order, 1, 8.99, Date
Order 1, 7.99, Date

3.4 How Tenants influence the Sizing of the Database Service

3.4.1 Database Connections

There is a database connection pool maintained per tenant. **The default is 20 connections for each tenant.**

The connection pool size is made up of:

- 10 connections for AIR Runtime
- 10 connections for Pricing application

Jobs that are ready to run wait for database connections for a limited time (30 seconds). If they don't get a connection, they are put back to queue.

You can change the **Pricing application pool size** in the database configuration dialog with the parameter "**connection pool size**". If there is 0 or no value, the default is used: 10

For sizing your connection pool size, you should take into account that each background job (like importer, exporter) uses about **two connections per job**. That means for larger load on one node, you need to set more connections.

If your pricing application is running in cloud mode, it makes no sense to change the settings in the database connection dialog. The settings are saved to the docker container's disk and are deleted if you restart the container (e.g. after update).

You have to use the environment variables for that. Please see the Operation Guide for details.

4 Scaling Options

4.1 Scaling Options for a Single Node

4.1.1 Controlling Concurrency of Jobs

For most efficient use of multiple CPU cores you can control the concurrency of jobs, i.e. how many jobs are running at the same time. A job is a unit of work that is executed in the background. Currently we support these types of jobs:

- Import Data
- Export Prices

- Optimize Prices
- Simulation

There is a job scheduler that controls the parallel execution of jobs. A job is not limited in the number of threads it is using. For instance, some importers use two or more threads for optimizing resource consumption.

To control the parallel execution there is a job concurrency setting located in the database connection settings dialog named "**job count**".

By default this is set to 4, which means the current node is able to execute 4 jobs at the same time.

Depending on your CPU count of the node, you can increase this setting. **We recommend 4 for a 2 core CPU system and 8 for a 4 core CPU system.**

In any case, please evaluate if the setting fits to your system and resource needs.

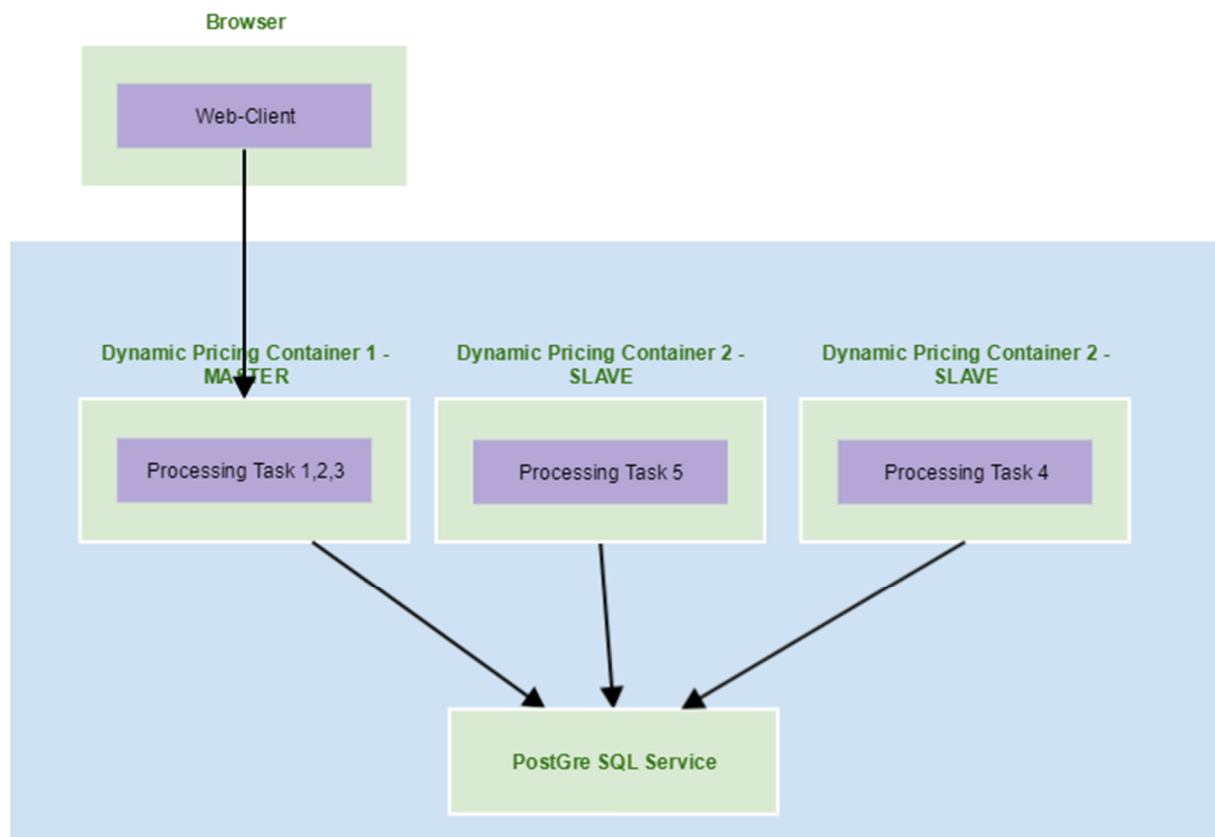
Job limitations per type:

Job Type	Multithreading approach	Number of simultaneously executable tasks
Importer	Mixed: <ul style="list-style-type: none"> • Single threaded (e.g. exchange rates, category data) • Dual threaded (e.g. transactions, item data) 	One importer per business unit Exception: Transaction Log importers: one per tenant
Computation	Single threaded	As many as job concurrency allows
Simulation	2 Threads	Only one per segment
Exporter	Single threaded	As many as job concurrency allows

If a job can't be scheduled, the AIR runtime handles this by queueing the jobs until a free resource slot is available

4.2 Horizontal Scaling with more Nodes

If one node is not enough for your resource needs then you can horizontally scale out by using more nodes. Dynamic pricing supports so called compute nodes (or "slave"). A compute node is the same like a normal application container (master) but does not respond to UI requests. Compute nodes just process jobs. The synchronisation is done by the database. A compute node asks regularly for unprocessed jobs and tries to register for them. If this works, then the node processed the job. A node regularly has to update the job table to signal that it is still alive. If the node dies, the other nodes are able to reclaim the job after some seconds again. Then the task is executed on a different node.



■ The system is not limited in the number of compute nodes.

See the Operation Guide about the configuration of such a compute node.

5 Performance Figures of the Reference System

5.1 Reference System Hard- and Software

5.1.1 Hardware and Software

Application Server	
CPU	4 cores, 2 GHz
Disk Storage	69 GB
RAM	10 GB
Java Runtime Max. Heap	1 GB
Docker	v.18.09.1
Java Runtime	OpenJDK 64-Bit Server (Zulu 8.33.0.1-linux64)

Database	
CPU	4 cores, 2.00 GHz
Disk Storage	48 GB
RAM	10 GB
PostgreSQL	9.6

5.2 Example for Resource Consumption on the Reference System

This is the example scenario we used for measurement:

Data Type	Count
Business Units	1
Items	50.000
Categories	100
Transactions (orders)	100.000
Competitor prices / item	2
Price computations	50.000

Process	Process Details	Time (HH:MM:SS)	Application Heap Memory (peak usage)	Database Memory (peak increase)	CPU usage (Percent of one core)	Application Disk Storage (temporary)	Database Storage	Bandwidth	Database Connections
Import	CSV Masterdata Import of 50000 products within 100 categories	00:02:15	142 MB (Heap)	+61 MB	Database: 100% Application: 20%	Application: +16 MB	Database: +444 MB	Import file size: 2,2 MB	12
Import	CSV Transaction Import of 100000 orders	00:01:03	146 MB (Heap)	+31 MB	Database: 100% Application: 20%	Application: +8 MB	Database: +30 MB	Import file size: 2,5 MB	12
Import	CSV Competitor Prices Import of 50000 prices from 2 competitors	00:07:03	238 MB (Heap)	+13 MB	Database: 100% Application: 5%	Application: +8 MB	Database: +48 MB	Import file size: 1,2 MB	13
Price Optimization	Computation of 50000 prices	00:06:33	470 MB (Heap)	+74 MB	Database: 40% Application: 100%	Application: +30 MB	Database: +14 MB	No network traffic	14
Simulation	Simulation of 5 days with 2 strategies(Support vector machine) with 50000 prices per day and strategy	01:05:15	527 MB (Heap)	+260 MB	Database: 40% Application: 100%	Application: +30 MB	Database: +84 MB	No network traffic.	14
Export	Optimized Price SAP XML Export of 50000 prices	00:00:46	410 MB (Heap)	+62 MB	Database: 100% Application: 20%	None	Application: +0.5 MB	Export file size; 39 MB	12
Export	Optimized Price CSV Export of 50000 prices	00:00:01	163 MB (Heap)	+119 MB	Database: 100% Application: 20%	None	Application: +15 MB	Export file size; 4 MB	12

Size figures are rounded to full megabytes.

CONTACT

GK Software SE
Waldstraße 7
08261 Schöneck
Germany

T +49 (0) 3 74 64 84 – 0
F +49 (0) 3 74 64 84 – 15
documentation@gk-software.com
www.gk-software.com