ALIBABA CLOUD

Alibaba Cloud

Tablestore

Pricing

Document Version: 20211110

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Billing overview

Tablestore charges resources for each instance. This topic describes the pricing, billing methods, and billing items of Tablestore.

Pricing

For more information about pricing, see Tablestore Pricing.

Billing methods

Tablestore charges resources based on the pay-as-you-go billing method. This method indicates that Tablestore charges resources you have used on an hourly basis.

Billing items

The following table lists the billing items of Tablestore.

Billing item		Billing method
Storage usage		The total storage usage charged based on the average storage usage per hour in a billing cycle.
Read throughput	Reserved read throughput	The total reserved read CUs of all tables charged based on the average reserved read CUs per hour in a billing cycle. **Note** Reserved read CUs are available only for high-performance instances. For more information about instances, see Instance.
	Additional read throughput	The total additional read CUs of all tables charged based on the average additional read CUs per second in a billing cycle.
Write throughput	Reserved write throughput	The total reserved write CUs of all tables charged based on the average reserved write CUs per hour in a billing cycle. **Note** Reserved write CUs are available only for high-performance instances. For more information about instances, see Instance.
	Additional write throughput	The total additional write CUs of all tables charged based on the average additional write CUs per second in a billing cycle.

Billing item		Billing method
Internet outbound traffic		The total Internet outbound traffic generated when applications access Tablestore over HTTP. **Note** • Tablestore charges Internet outbound traffic instead of inbound traffic and internal network traffic. • When access fails, Tablestore returns access failure information, which also generates outbound traffic. • Access between different regions is also billed as Internet access.

Note

- You are also charged when you use search indexes. For more information about the billing, see Billable items of search indexes.
- You are also charged when you use secondary indexes. For more information about the billing, see Billing rules.

2.Storage usage

Tablestore charges storage usage on an hourly basis based on the total size of data. Tablestore calculates the total size of data based on predetermined intervals and calculates the average data size per hour.

> **Note** For more information about pricing, see Tablestore pricing.

The following section describes how to calculate the size of a single row and tables.

Calculate the size of a single row

Each row of data in Tablestore uses storage space. After max versions or time to live (TTL) is configured, data of each version includes the version number (8 bytes), column name, and data value.

The size of a single row is calculated based on the formula: **Size of a single row = Size of primary key columns + Size of all attribute columns. For more information, see** Primary keys and attributes.

The sizes of primary key columns and all attribute columns are calculated based on the formulas:

- Size of primary key columns = Length of all primary key column names + Size of values in primary key columns
- For more information about the size of all attribute columns, see the examples on how to calculate the size of a single row and a table in this topic.

The following table describes how to calculate the size of values.

Date type	Number of bytes
STRING	The number of bytes used by characters encoded in UTF-8. Tablestore allows empty strings. The size of empty strings is 0.
INTEGER	8.
DOUBLE	8.
BOOLEAN	1.
BINARY	The number of bytes used by binary data.

Example: Calculate the size of a row.

The primary column name is ID. Other columns are attribute columns.

ID	Name	Length	Comments
1.	timestamp = 1466676354000, value = 'zhangsan'	timestamp = 1466676354000, value = 20	timestamp = 1466676354000, value = String (100 Bytes) timestamp = 1466679954000, value = String (150 Bytes)

In the row, two versions are available for the value in the Comments column.

- When MaxVersions is set to 2 and TTL is set to 2592000:

  The size of an attribute column is calculated based on the formula: Size of an attribute column = (Length of the attribute column name + 8) × Number of valid versions + Total size of values in the attribute column.

  > **Note** When max versions is set to a value greater than 1 or TTL is set to -1, each version number uses 8 bytes. The version number is represented by a timestamp. For more information, see Data versions and time to live.

  The size of the row is the sum of 10, 20, 22, and 282. Details:

  - Size of the primary key column: len ('ID') + len (1) = 10 bytes
  - Size of the Name attribute column: (len ('Name') + 8) * 1 + len ('zhangsan') = 20 bytes
  - Size of the Length column: (len ('Length') + 8) * 1 + len (20) = 22 bytes
  - Size of the Comments column: (len ('Comments') + 8) * 2 + 100 + 150 = 282 bytes

- When MaxVersions is set to 1 and TTL is set to -1:

  The size of an attribute column is calculated based on the formula: Size of an attribute column = Length of the attribute column name + Total size of the value in the attribute column.

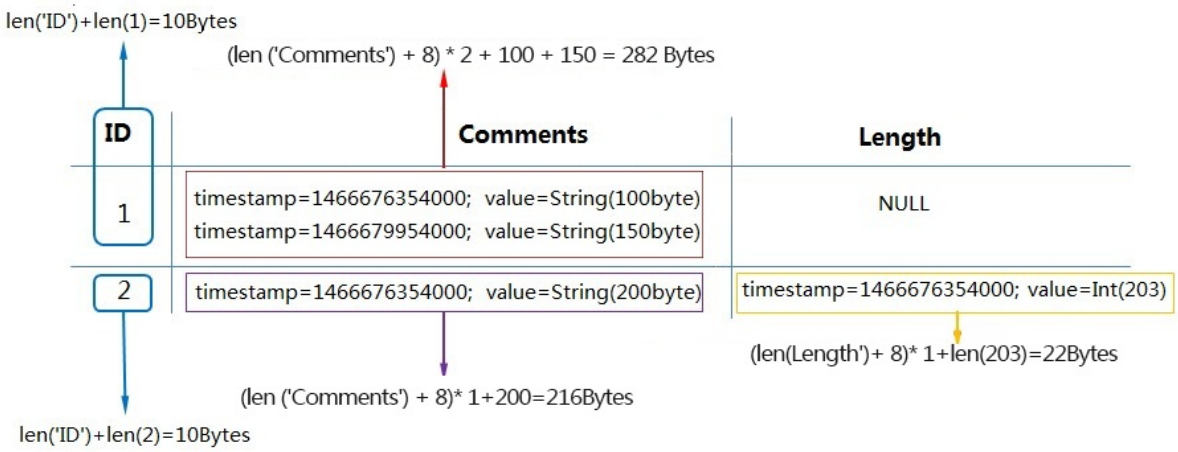
  > **Note**
  > - When MaxVersions is set to 1 and TTL is set to -1, the version number consumes no bytes.
  > - Although the value in the Comments column has two versions, only the latest version is calculated because MaxVersions is set to 1.

  The size of the row is the sum of 10, 12, 14, and 158, which equals 194 bytes. Details:

  - Size of the primary key column: len ('ID') + len (1) = 10 bytes
  - Size of the Name attribute column: len ('Name') + len ('zhangsan') = 12 bytes
  - Size of the Length attribute column: len ('Length') + len (20) = 14 bytes
  - Size of the Comments attribute column: len ('Comments') + 150 (bytes) = 158 bytes

Calculate the size of a table

The size of a table consists of sizes of all rows of data in the table. Example: A table contains the ID primary key column and attribute columns. When MaxVersions is set to 2 and TTL is set to -1: The following figure shows how to calculate the size of the table.



- The size of the row whose value in the ID column is 1 = 10 (size of the primary key column) + 282 (size of the values of the two versions in the Comments column). In total, the size of the row whose value in the ID column is 1 is 292 bytes.
- The size of the row whose value in the ID column is 2 = 10 (size of the primary key column) + 216 (size of the value in the Comments column) + 22 (size of the value in the Length column). In total, the size of the row whose value in the ID column is 2 is 248 bytes.
- The size of the table is the sum of 292 and 248, which equals 540 bytes.

If the size of the data in the table remains unchanged within one hour, Tablestore charges storage fees based on 540 bytes. Tablestore does not impose limits on the size of data that can be stored in a table. Tablestore charges storage fees for actual data stored in the table.

> **Note** Tablestore asynchronously deletes expired data in each partition and the additional versions when the max versions value is exceeded. The time used to delete data is related to the total size of the data, which is within 24 hours. Data written to a partition after a data delete operation is performed is charged after the next data delete operation is performed.

3.Billable items of search indexes

This topic describes the billable items and billing formulas of search indexes. Search index-based queries require extra space to store indexed data and consume read throughput.

> **Note**
> - Billable items of indexes are independent of data tables.
> - The price of each billable item of a search index is the same as that of a high-performance instance.

Billable items

Billable item	Billing method	Description
Data size	Pay-as-you-go	Unit: GB. Billed size is rounded up to the next GB. Tablestore charges you for the total volume of indexed data on an hourly basis. The utilization of system resources varies with field types and index types. Therefore, indexed data is billed based on the volume of data compressed after you create indexes. The raw data volume in a data table does not affect this billing.

<table>
<tr><th colspan="2">Billable item</th><th>Billing method</th><th>Description</th></tr>
<tr><td rowspan="2">Read throughput</td><td>Reserved Read Throughput</td><td>Pay-as-you-go</td><td>Unit: CU.

Tablestore specifies a reserved read throughput based on the indexed data size. The charges of reserved read throughput are calculated based on the following operations:

• When you create a search index, Tablestore reads data from a data table, which consumes read throughput.
• Tokenization during the creation of a search index also consumes read throughput. Fees incurred from tokenization are included in the fees for the reserved read throughput.
• To ensure index and query performance, some indexed data is loaded to the memory in advance and remains in memory. Such data consumes read throughput. Fees incurred from these operations are also included in the fees for the reserved read throughput.

The minimum fees are based on the reserved read throughput value. For example, assume that the reserved read throughput is 10,000 CUs for an index. Each index query reads 10 rows, and the size of each row is less than 4 KB. When the number of queries per second (QPS) is less than 1,000, the actual consumed read throughput is less than the reserved read throughput, and no extra fees are charged.

Calculation of reserved read throughput: The reserved read throughput is proportional to the size and number of rows of the indexed data. For example, 1 GB or 2 million rows of indexed data corresponds to a reserved read throughput of 10 CUs. When the reserved read throughput values corresponding to the data size and the number of rows are different, the system uses the larger one as the reserved read throughput.

Note
• The maximum reserved read throughput for Tablestore is 100,000 CUs.
• When the data size is less than 200 MB and the number of rows is less than 400,000, the reserved read throughput for Tablestore must be 20 CUs, which is suitable for tests that involve a small amount of data. When the data size is greater than or equal to 200 MB or the number of rows is greater than or equal to 400,000, the reserved read throughput must be greater than 100 CUs.</td></tr>
<tr><td>Additional throughput</td><td>Pay-as-you-go</td><td>The portion of the actual consumed read throughput that exceeds the reserved read throughput is charged as additional read throughput.

Unit: CU.</td></tr>
</table>

Billable item	Billing method	Description
Internet outbound traffic	Pay-as-you-go	Fees for Internet outbound traffic. Unit: GB.

Billing formulas

The following table describes how to calculate the data size and read throughput of a search index.

Billable item	Formula	Description
Data size	$f(Size) = \frac{Size}{1GB} * USD\ 0.00030/GB/Hour$	The Size variable indicates the size of the compressed indexed data.
Read throughput	Reserved read throughput per index: $ReservedCuPerIndex(Size, Rows) = min(max(\frac{Size}{0.2GB}, \frac{Rows}{0.4m}, 10) * 2, 100000)$ Read throughput per query: $CuPerQuery = Ceil(\frac{ReturnRowSize}{4KB}) * ReturnRowCount$	• The Size variable indicates the size of the compressed indexed data. • The Rows variable indicates the total number of rows in an index, not including the child rows in nested queries. • The ReturnRowSize variable indicates the size of returned rows. • The ReturnRowCount variable indicates the number of returned rows.

Billing examples

Storage	Number of rows	Billing
8 GB	9 million	• Storage fees: 8 GB × USD 0.00030/GB/Hour = USD 0.0024/Hour • Calculation of reserved read throughput: 8 GB of data corresponds to 80 CUs and 9 million rows correspond to 45 CUs. Fees of reserved read throughput: 80 CUs × USD 0.0002/CU/Hour = USD 0.016/Hour • Total fees: USD 0.0024/Hour + USD 0.016/Hour = USD 0.0184/Hour The portion of actual consumed read throughput that exceeds the reserved read throughput is charged as additional read throughput. The billing method of Internet outbound traffic is the same as that of the data table.

Storage	Number of rows	Billing
100 GB	300 million	• Storage fees: 100 GB × USD 0.00030/GB/Hour = USD 0.03/Hour • Calculation of reserved read throughput: 100 GB of data corresponds to 1,000 CUs and 300 million rows correspond to 1,500 CUs. The reserved read throughput is 1,500 CUs. Fees of reserved read throughput: 1,500 CUs × USD 0.0002/CU/Hour = USD 0.3/Hour • Total fees: USD 0.03/Hour + USD 0.3/Hour = USD 0.33/Hour The portion of actual consumed read throughput that exceeds the reserved read throughput is charged as additional read throughput. The billing method of Internet outbound traffic is the same as that of the data table.
30 TB	10 billion	• Storage fees: 30,000 GB × USD 0.00030/GB/Hour = USD 9/Hour • Calculation of reserved read throughput: 30 TB of data corresponds to 300,000 CUs and 10 billion rows correspond to 50,000 CUs. Although the former CU value is larger, it exceeds the maximum reserved read throughput of 100,000 CUs. Therefore, the reserved read throughput is 100,000 CUs. Fees of reserved read throughput: 100,000 CUs × USD 0.0002/CU/Hour = USD 20/Hour • Total fees: USD 9/Hour + USD 20/Hour = USD 29/Hour The portion of actual consumed read throughput that exceeds the reserved read throughput is charged as additional read throughput. The billing method of Internet outbound traffic is the same as that of the data table.

Note The price described in the preceding table is for reference only. For more information, see the Tablestore console.

4.Billable items of secondary indexes

Additional storage usage is required to use secondary index. When you inserts data to a data table, the system may also need to write the index tables created on the data table. During this process, read and write CUs are consumed. This topic describes the billing rules for secondary index.

> **Note** Capacity units (CUs) are read and write throughput units. They are the smallest units used to measure the costs of read and write operations. For example, when the system reads 4 KB from one row per second, one read CU is consumed.

To use secondary indexes, index tables are needed. Therefore, additional storage space is required to store index tables. When the system inserts data to a data table, it may also need to write the index tables created on the data table at the same time. During this process, read and write CUs are consumed.

Secondary index billing includes the following parts: the number of read and write CUs consumed to write index tables, the amount of data stored in the index tables, and the amount of data that is read from the index tables.

Billable item	Description
Data storage	The storage space used to store a data table and its index tables.
Read CUs consumed to write index tables	The number of CUs that are consumed by read operations to delete, insert, or update index rows.
Write CUs consumed to write index tables	The number of CUs that are consumed to insert or update index rows.
CUs consumed by regular read operations	The number of CUs that are consumed to read data from a data table or index tables by using an API.
CUs consumed by regular write operations	The number of CUs that are consumed to insert data to a data table by using an API.

Billing rules for storing, writing, and reading an index table:

- The billing rules for storing and reading an index table are the same as those of a data table. For more information, see Billing overview.
- CUs are consumed based on the following rules when the system writes an index table:
  - Write CUs are consumed only when an index row is inserted or updated.
  - Read CUs are consumed when an index row is deleted, updated, or inserted. The number of read CUs equals the amount of data read from the corresponding indexed columns in the data table.

Calculate the number of read CUs consumed to write index tables

When you create secondary indexes on the data table, read CUs are consumed based on the following rules:

- When you use the PUT operation to insert a data row to the data table:

- The PUT operation does not insert data to the indexed attribute columns in the data table, which means that no index row is inserted. In this case, one read CU is consumed.
- The PUT operation inserts data to the indexed attribute columns in the data table, which means that new index rows are inserted. In this case, one read CU is consumed.

- When you use the PUT operation to overwrite a row in the data table:
  - The PUT operation does not update the indexed attribute columns in the data table. In this case, one read CU is consumed.
  - The PUT operation updates the indexed attribute columns in the data table. In this case, the read CUs are consumed:

    Divide the total amount of data read from the indexed attribute columns by four, excluding primary key columns. The number of consumed CUs equals the calculated value rounded up to the nearest integer. If the total amount is 0 KB, one CU is consumed.
- When you use the UPDATE operation to insert a data row to the data table:
  - If the UPDATE operation does not insert data to the indexed columns in the data table, no read CU is consumed.
  - If the UPDATE operation inserts data to the indexed columns in the data table, one read CU is consumed.
- When you use the UPDATE operation to update a row in the data table:
  - If the UPDATE operation does not insert data to the indexed attribute columns in the data table, no read CU is consumed.
  - If the UPDATE operation inserts data to the indexed attribute columns in the data table, read CUs are consumed based on the following rules:

    Divide the total amount of data read from the indexed columns by four, excluding the primary key columns. The number of consumed CUs equals the calculated value rounded up to the nearest integer. If the total amount is 0 KB, one CU is consumed.
- When you use the Delete operation to delete a row in the data table, read CUs are consumed based on the following rules:

  Divide the total amount of data read from the indexed columns by four, excluding the primary key columns. The number of consumed CUs equals the calculated value rounded up to the nearest integer. If the total amount is 0 KB, one CU is consumed.
- If the data table uses primary key auto increment, inserting data to the data table does not consume any read CUs. Updating a row in a data table that uses primary key auto increment consumes read CUs. CUs are calculated based on the same rules as those of the UPDATE operation.

  > **Note** We recommend that you use primary key auto increment to insert data to a data table to decrease the number of CUs that are consumed by index tables.
  >
  > For data tables that do not use primary key auto increment, one read CU is consumed if a read operation is performed on the indexed columns, even if no data is retrieved. For data tables that use primary key auto increment, no read operation is performed on the indexed columns when you insert data. Therefore, no read CU is consumed.

Calculate the number of write CUs

When you insert data to the data table and create secondary indexes, write CUs are consumed. Write CUs are consumed based on the following rules:

- If you insert a row to the data table and no data in the index table is updated, no write CUs are consumed.
- If you insert a row to the data table and a new index row is inserted to the index table, write CUs are consumed. The number of the write CUs is determined by the size of the inserted index row.
- If you insert a row to the data table and an index row is deleted from the index table, write CUs are consumed. The number of the write CUs is determined by the size of the deleted index row.
- If you insert a row to the data table and an index row in the index table is updated, write CUs are consumed. The number of the write CUs is determined by the size of the updated index row.
- If you insert a row to the data table, an index row is deleted from the index table, and another index row is inserted to the index table, write CUs are consumed. The number of the write CUs is determined by the total size of the deleted and inserted index rows.

Detailed rules:

- When you use the PUT operation to insert a data row to a data table:
  - The PUT operation does not insert data to the indexed attribute columns in the data table, which means that no index row is inserted. In this case, no read CU is consumed.
  - The PUT operation inserts data to the indexed attribute columns in the data table, which means that new index rows are inserted. The write CUs consumed for each index table:

    Divide the total amount of data in the inserted index row by four. The number of consumed CUs equals the calculated value rounded up to the nearest integer.
- When you use the PUT operation to overwrite a row in the data table:
  - The PUT operation only updates the indexed primary key columns in the data table. In this case, no write CUs are consumed.
  - The PUT operation updates the indexed columns in the data table. The write CUs are consumed based on the following rules:

    All indexes updated by the PUT operation consume a certain number of write CUs, except sparse indexes.
- When you use the UPDATE operation to insert a data row to the data table:
  - If the UPDATE operation does not insert data to the indexed columns in the data table, no write CUs are consumed.
  - If the UPDATE operation inserts data to the indexed columns in the data table, the write CUs consumed for each index table:
    - If the UPDATE operation inserts a new index row, write CUs are consumed. Divide the total size of the data in the index row by four. The number of consumed CUs equals the calculated value rounded up to the nearest integer.
    - If no index row is inserted, no write CUs are consumed.
- When you use the UPDATE operation to update a row in the data table:
  - If the UPDATE operation does not update the indexed attribute columns, no write CUs are consumed.

- If the UPDATE operation updates the indexed attribute columns, write CUs consumed for each index table are calculated based on the following rules:
  - If the index table already contains an index row created based on the row to be updated, delete CUs are consumed. The number of the delete CUs is determined by the size of the indexed primary keys in the deleted index row.
  - If a new index row is inserted based on the updated row, write CUs are consumed. The number of the write CUs is determined by the size of the indexed primary keys in the inserted index row.
  - If the UPDATE operation only updates the attribute data in the existing index row but no new index row is inserted, update CUs are consumed.

  Divide the total amount of data in the index row by four. The number of consumed CUs equals the calculated value rounded up to the nearest integer.

- When you use the DELETE operation to delete a row in the data table, write CUs are consumed based on the following rules:

  If an index table already contains an index row created based on the row to be deleted, write CUs are consumed. Divide the total amount of the data in the corresponding indexed columns by four, excluding the primary key columns. The consumed write CUs equal the calculated value rounded up to the nearest integer.

- If you insert data to a data table that uses primary key auto increment, write CUs are consumed. The write CUs are calculated based on the same rules as those of the PUT operation. If you update a row in a data table that uses primary key auto increment, write CUs are consumed. The write CUs are calculated based on the same rules as those of the UPDATE operation.

Measure index table size

The size of an index table is measured based on the same rule as that of a data table. The size of an index table equals the total size of all rows. The total size of the rows equals the total size of primary keys and attribute data. For more information, see Data storage.

Calculate the number of CUs consumed to read an index table

When you use an SDK, the console, or other methods, such as a DLA, to read an index table, read CUs are consumed. The number of read CUs are calculated based on the same rules as those of reading a data table.

Examples

The following example uses a data table that has two index tables to describe how CUs are consumed under different conditions.

The data table Table contains two primary key columns PK0 and PK1, and three predefined columns Col0, Col1, and Col2. Two index tables, Index0 and Index1, are created on the data table. Index0 contains three primary keys Col0, PK0, and PK1 and one attribute column Col2. Index1 contains four primary keys Col1, Col0, PK0, and PK1, and no attribute columns. Use the UPDATE operation to update PK0 and PK1.

- If the row does not exist in the data table:
  - Updating Col3 does not consume read or write CUs.
  - Updating Col1 consumes the following CUs:
    - One read CU
    - No write CUs

- Updating Col0 and Col1 consumes the following CUs:
  - One read CU
  - Index0 consumes write CUs. The number of the write CUs is determined by the total amount of data inserted to Col0, PK0, and PK1. Index1 consumes write CUs. The number of the write CUs is determined by the total amount of data inserted to Col0, Col1, PK0, and PK1.

- If the row already exists in the data table:
  - Updating Col3 does not consume read or write CUs.
  - Updating Col2 consumes the following CUs:
    - Read CUs are consumed. The number of the read CUs is determined by the amount of data read from Col0. If the UPDATE operation inserts data to Col0, one CU is consumed.
    - For Index0, if the UPDATE operation insets data to Col0, Index0 does not consume write CUs. If the UPDATE operation updates the data in Col0, Index0 consumed write CUs. The number of the write CUs is determined by the total amount of data inserted to Col0, PK0, PK1, and Col2. Index1 does not consume write CUs.
  - Updating Col1 consumes the following CUs:
    - Read CUs are consumed. The number of the read CUs is determined by the amount of data read from Col0 and Col1. If the total amount is 0 KB, one CU is consumed.
    - Index0 does not consume write CUs. For Index1, if an index row is inserted, write CUs are consumed. The number of the write CUs is determined by the amount of data read from Col0 and inserted to Col1, PK0, and PK1. For Index1, if no data in Col0 is updated, no index row is inserted and no write CUs are consumed. If the data in Col0 and Col1 is updated, write CUs are consumed to delete the corresponding index row. The number of write CUs is determined by the total amount of data read from Col0, Col1, PK0, and PK1.

5.Overdue payments, renewals, and upgrades

This topic describes the overdue payment, renewal, and upgrade policies of Tablestore instances.

> **Notice** You may receive notifications if you have overdue payments. When this occurs, pay off all overdue bills to avoid instances being released. Note that your instances may be released at a system-selected time after the payment due date.

Billing method	Expiration or overdue payment	Renewal and upgrade
Pay-as-you-go	Fees are calculated on an hourly basis. When your account balance is insufficient to cover the charges of the last billing cycle, you have an overdue payment. When an overdue payment is generated, the system sends notifications to you based on the following situations: • You are not affected by the service suspension if you top up your balance within 24 hours. • Your Tablestore is automatically suspended if you fail to pay off all overdue bills within 24 hours. However, you are still charged for the storage space resources that are used. Consequently, the overdue amount continues to increase. Your Tablestore is automatically started if you top up your balance to pay off all overdue bills within 15 days after Tablestore is suspended. • If you fail to pay off all overdue bills within 15 days, you are regarded as voluntarily discarding Tablestore. Data in your Tablestore instance may be deleted and deleted data cannot be recovered.	Pay-as-you-go instances are charged based on the service duration. You do not need to renew the instances. Instead, you need only to top up your balance in the Alibaba Cloud Management console.

6.Billing examples

This topic provides billing examples to describes how Tablestore calculates fees.

Background information

A user in the United States creates a high-performance instance after they activate Tablestore. Data in the table of the instance supports queries per second (QPS) of 10000 and throughput of smaller than 4 KB (1 CU). The user wants to learn about how Tablestore calculates fees related to the table within one day.

Case analysis

> **Note** The unit price for Tablestore was released on the Alibaba Cloud official website on August 1, 2018. To remain updated on the unit price, log on to the Alibaba Cloud official website.

Billing item	Unit price for high-performance instances
Additional read throughput	USD 0.0030 per 10,000 CUs

Additional read throughput within the day is charged based on the formula:

Read throughput fees = 10000 × 86400/10000 × 0.0030. A total of USD 259.2 is charged.

> **Note** When Tablestore calculates additional read/write throughput fees, the total number of consumed CUs is calculated. In this example, the total number of consumed CUs is 864 million CUs (10000 × 86400).

7.FAQ

7.1. What is reserved read/write throughput?

Reserved read/write throughput is an attribute of data tables for high-performance instances. The system reserves resources based on the reserved read/write throughput settings for data tables to meet the throughput demands of the data tables.

When you call the CreateTable operation to create a data table for a high-performance instance, you must specify the reserved read/write throughput. After the data table is created, you can call the UpdateTable operation to modify the reserved read/write throughput settings of the data table.

The reserved read/write throughput is measured by capacity units (CUs). You can set the number of reserved read CUs or write CUs for a data table to a value greater than or equal to 0. By default, the maximum value is 100,000. If you need more than 100,000 reserved read CUs or write CUs, you can submit a ticket to increase the value.

If the number of reserved read CUs or write CUs is greater than 0, you are charged even if no read or write request is sent to the data table.

Read CUs and write CUs are consumed when you call Tablestore API operations to read and write data.

Tablestore generates bills on an hourly basis based on the sum of reserved read CUs and write CUs for all data tables in the high-performance instance. The specified reserved read/write throughput settings may be modified. Tablestore collects statistics about the reserved read CUs and write CUs for data tables at a specific interval, calculates the average number of reserved read CUs and write CUs per hour, and generates bills by multiplying the average number of reserved read CUs and write CUs per hour by their respective unit prices. The unit price of the reserved read CUs or write CUs may change. For more information, see the Pricing tab of Tablestore.

7.2. How does Tablestore charge storage fees?

Tablestore charges data storage fees on an hourly basis. The total size of data stored in Tablestore tables changes in real time. Tablestore collects statistics on the total size of data stored in Tablestore tables at pre-determined intervals, calculates average total size per hour, and calculates fees by multiplying the average total size per hour by the unit price. The unit price may change. For more information about pricing, see Tablestore Pricing.

The total data size of an instance indicates the size of the data stored in all tables of the instance. The table size is the total size of all rows of data in the table. The following example describes how to calculate the size of data in a table.

Example: A table contains the id primary key column and the name, length, and comments columns.

id	name	length	comments
Integer(1)	String(10byte)	Integer	String(32Byte)
Integer(2)	String(20byte)	Integer	String(999Byte)
Integer(3)	String(43byte)	Integer	空

- The size of the row whose value in the id column is 1 is calculated based on the following formula: Size of the row whose value in the id column is 1 = len ('id') + len ('name') + len ('length') + len ('comments'). In other words, size of the row whose value in the id column is 1 is the sum of 8, 10, 8, and 32 bytes, which equals 58 bytes.
- Size of the row whose value in the id column is 2 is calculated based on the following formula: Size of the row whose value in the id column is 2 = len ('id') + len ('name') + len ('length') + len ('comments'). In other words, size of the row whose value in the id column is 2 is the sum of 8, 20, 8, and 999, which equals 1,035 bytes.
- Size of the row whose value in the id column is 3 is calculated based on the following formula: Size of the row whose value in the id column is 3 = len ('id') + len ('name') + len ('length'). In other words, size of the row whose value in the id column is 3 is the sum of 8, 43, and 8, which equals 59 bytes.
- The table size is the sum of 58, 1035, and 59, which equals 1,152 bytes.

If the table size remains unchanged within one hour, fees are calculated based on 1,152 bytes. Tablestore does not impose limits on the amount of data that can be stored in a single table. You can store as much data as necessary. You are charged for the resources you use.

7.3. What are billing methods and items of Tablestore?

Tablestore is a pay-as-you-go service. Fees are calculated based on the resources used each billing cycle. No additional fees are charged when you buy instances.

Tablestore provides the following billing items:

- Storage usage
- Reserved read/write throughput
- Additional read/write throughput
- Internet outbound traffic

Fees are calculated on an hourly basis based on the actual usage of resources when you use Tablestore.

Example 1: Calculate the fees of resources used during one hour

- Scenario

  During a billing cycle of one hour: Storage usage is 50 GB, Internet outbound traffic usage is 10 GB, and the reserved read/write throughput settings are changed from (1000,1500) to (1200,800) at the 20th minute. A total of 50,000 additional read capacity units (CUs) and 10,000 additional write CUs are used within the billing cycle.

- Formulas

Formulas to calculate the fees within the one hour:

- Storage fees = 50 GB × Unit storage price/GB
- Traffic fees = 10 GB × Unit price of Internet outbound traffic/GB
- Fees for reserved read and write CUs:
  - Average reserved read CUs: (1000 × 20 + 1200 × 40)/60 = 1133.3
  - Average reserved write CUs: (1500 × 20 + 800 × 40)/60 = 1033
  - Total fees of reserved read and write CUs: 1133.3 × Unit price of reserved read CU/hour + 1033.3 × Unit price of reserved write CU/hour
- Total fees of additional read and write CUs: 50000/10000 × Unit price of additional read CU per 10000 CUs + 10000/10000 × Unit price of additional write CU per 10000 CUs

The total fees consist of the storage fees, traffic fees, fees of reserved read and write CUs, and fees of additional read and write CUs.

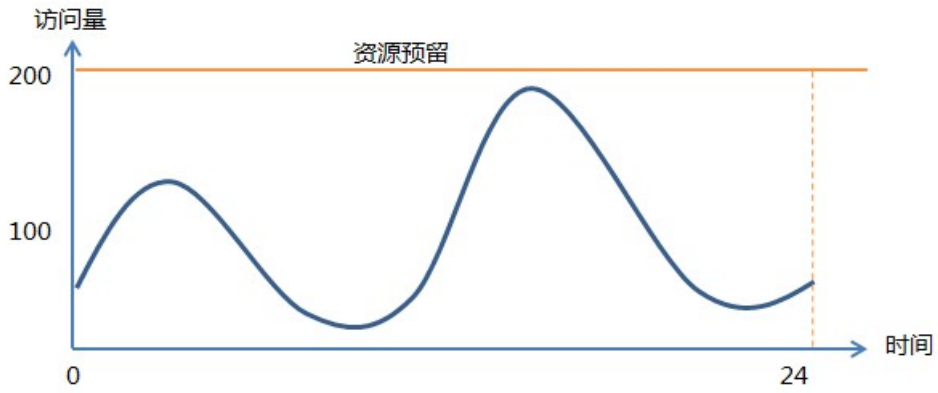
The billing of storage usage and reserved read and write CUs is accurate to the minute. The average storage usage and reserved read and write CUs apply when the system calculate fees at the end of a billing cycle. The billing of additional read and write CUs is accurate to the second. Fees are calculated based on used CUs in each second.

Example: Within the first 20 minutes, the reserved read throughput is set to 1000 CUs. During one second, 2,100 CUs are consumed. Additional read CUs are calculated based on the following formula: Additional read CUs = 2100 - 1000. This way, additional read CUs are 1100.

Example 2: Calculate the fees of resources used within one day

- Scenario 1

  The following section describes the billing methods that are used when you buy resources in a traditional way.



The preceding line chart simulates the access to an application in one day. Assume that the visits generated by read and write requests to the application are the same for the convenience of illustration. To ensure that sufficient resources are available for the application to provide read and write services during peak hours, you must buy resources based on your business at peak hours. In Tablestore, you can buy 200 CUs for each of reserved read throughput and write throughput.

Formulas

Total fees = 200 × Unit price of reserved read CU/hour × 24 + 200 × Unit price of reserved write CU/hour × 24 + Storage fees within 24 hours + Traffic usage within 24 hours × Unit price of Internet outbound traffic.

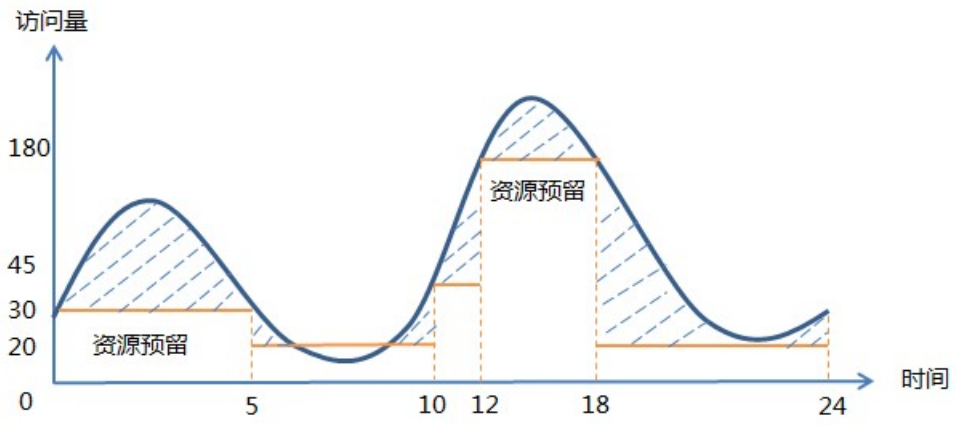
In other words, total fees = 4800 × Unit price of reserved read CU/hour + 4800 × Unit price of reserved write CU/hour + Storage fees within 24 hours + Internet outbound traffic within 24 hours.

Tablestore provides the operation to adjust the settings of reserved read/write throughput. **You can call the operation to adjust the reserved read and write CUs for each table anytime.** The adjustment takes effect within one minute and does not affect business.

You can use this method and increase reserved CUs to meet business requirements at peak hours and decrease the reserved CUs for off peak hours to minimize costs.

- Scenario 2

  Example: Specify different reserved read/write CUs based on different periods within a day.



  Read and write CUs consumed within the 24 hours:

  - 00:00:00-05:00:00: The reserved read throughput is set to 30 CUs. The reserved write throughput is set to 30 CUs. Within the five hours, the reserved read CUs are 100,000. The reserved write CUs are 100,000.
  - 05:00:00-10:00:00: Access to the application decreases. Decrease the reserved read CUs to 20 and reserved write CUs to 20. Within the five hours, additional read CUs are 5,000. Additional write CUs are 5,000.
  - 10:00:00-12:00:00: Access to the application increases. The reserved read CUs are increased to 45 and the write CUs are increased to 45. Within the two hours, additional read CUs are 10,000. Additional write CUs are 10,000.
  - 12:00:00-18:00:00: business peak hours. The reserved read CUs are increased to 180 and the reserved write CUs are increased to 180. Within the six hours, additional read CUs are 30,000. Additional write CUs are 30,000.
  - 18:00:00-24:00:00: business off peak hours. Decrease the reserved read CUs to 20 and write CUs to 20. Within the six hours, additional read CUs are 50,000. Additional write CUs are 50,000.

  Formulas

  To facilitate calculation, assume that consumed read CUs are the same as consumed write CUs and reserved read CUs are the same as reserved write CUs. The fees generated in a day are calculated as follows:

- Fees of consumed read CUs:

  Fees of consumed read CUs = (30 × 5 hours + 20 × 5 hours + 45 × 2 hours + 180 × 6 hours + 20 × 6 hours) × Unit price of reserved read CU/hour + (100000 + 5000+ 10000 + 30000 + 50000) × Unit price of additional read CU/hour

  In other words, fees of consumed read CUs = 1540 × Unit price of reserved read CU/hour + 195000 × Unit price of additional read CU/hour.

- Fees of consumed write CUs:

  Fees of consumed write CUs = (30 × 5 hours + 20 × 5 hours + 45 × 2 hours + 180 × 6 hours + 20 × 6 hours) × Unit price of reserved write CU/hour + (100000 + 5000 + 10000 + 30000 + 50000) × Unit price of additional write CU/hour

  In other words, fees of consumed write CUs= 1540 × Unit price of reserved write CU/hour + 195000 × Unit price of additional write CU/hour.

Compared with the method used to buy resources in Scenario 1, the method used in Scenario 2 is more cost-effective:

Saved fees = 4800 × Unit price of reserved read CU/hour + 4800 × Unit price of reserved write CU/hour - 1540 × Unit price of reserved read CU/hour - 19.5 × Unit price of additional read CU per 10000 CUs - 1540 × Unit price of reserved write CU/hour - 19.5 × Unit price of additional write CU per 10000 CUs

References

- The unit prices of reserved read and write CUs are higher than those of additional read and write CUs. We recommend that you adjust the reserved read and write CUs to minimize costs.
- You can use Tablestore SDK to set the reserved read and write CUs to relatively low values to minimize costs.