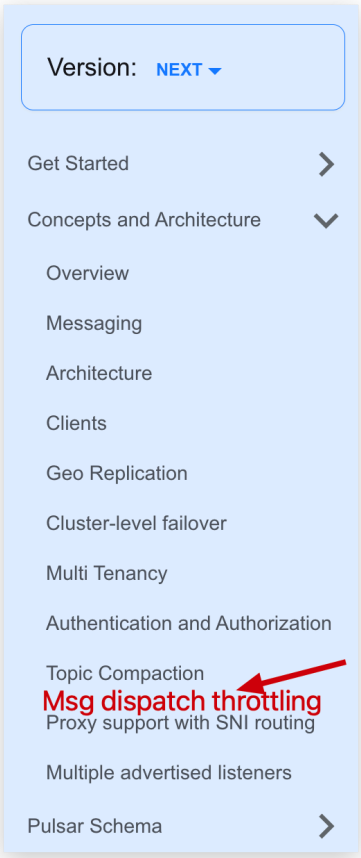
Original version: https://github.com/apache/pulsar/pull/19012.
New PR: https://github.com/apache/pulsar-site/pull/386.

Table of Contents

Navigation

I suggest adding a new topic in parallel with “topic compaction” under “Concepts”, which also provides chances for scalability when we want to add more sub-features about throttling.



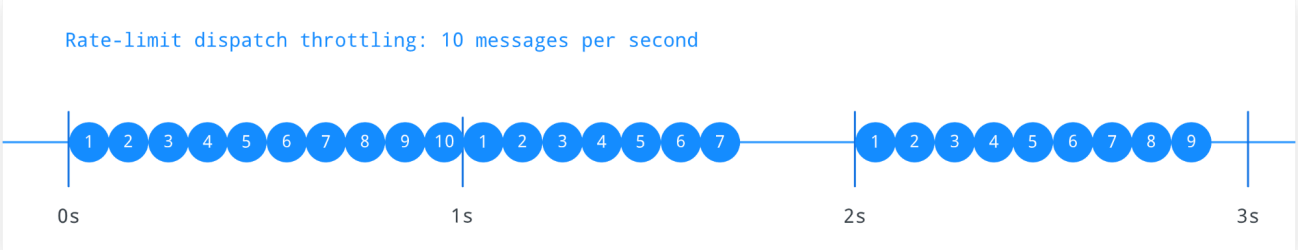
Original version: https://github.com/apache/pulsar/pull/19012.
New PR: https://github.com/apache/pulsar-site/pull/386.

Message dispatch throttling

Overview

What is message dispatch throttling?

Large message payloads can cause memory usage spikes that lead to performance decreases. Pulsar adopts a rate-limit throttling mechanism for message dispatch, avoiding a traffic surge and improving message deliverability. You can set a threshold to limit the number of messages and the byte size of entries that can be delivered to clients, blocking subsequent deliveries when the traffic per unit of time exceeds the threshold.
For example, when you configure the dispatch rate limit to 10 messages per second, then the number of messages that can be delivered to the client per second is up to 10.



Why use it?

Message dispatch throttling brings the following benefits in detail:

- **Limit broker's read request loads to BookKeeper**
  Messages are persistently stored in the BookKeeper cluster. If a large number of read requests cannot be fulfilled using the cached data, the BookKeeper cluster may become too busy to respond, and the broker's I/O or CPU resources can be fully occupied. Using the message dispatch throttling feature can regulate the data flow to limit the broker's read request loads to BookKeeper.
- **Balance the allocation of broker's hardware resources at topic/subscription levels**
  A broker instance serves multiple topics at one time. If a topic is overloaded with requests, it will occupy almost all of the I/O, CPU, and memory resources of the broker, causing other topics cannot be read. Using the message dispatch throttling feature can limit the allocation of broker's hardware resources across topics.
- **Limit the allocation of client's hardware resources at topic/subscription levels**
  When there is a large backlog of messages to consume, clients may receive a large amount of data in a short period of time, which monopolizes their computing resources. Since the client has no mechanisms to proactively limit the consumption rate, using the message dispatch throttling feature can also regulate the allocation of the client's hardware resources.

Original version: https://github.com/apache/pulsar/pull/19012.
New PR: https://github.com/apache/pulsar-site/pull/386.

How it works

The process of message dispatch throttling can be divided into the following steps:
1. The broker approximates the number of entries to read from the bookies by calculating the remaining quota.
2. The broker reads the messages from the bookies.
3. The broker dispatches the messages to the client and updates the counter to decrease the quota. A scheduled task refreshes the quota when a throttling period ends.

Note:

- The quota cannot be decreased before step 3, because the broker doesn't know the actual number of messages per entry or the actual entry size until it reads the data.
- Operations like `seek` or `redeliver` may deliver messages to a client multiple times. The broker counts them as different messages and updates the counter.

Limitations

Message dispatch throttling may cause messages over-delivered per unit of time due to the following reasons:

1. The broker may read more entries or bytes from the bookies than the throttling limit.

1) The byte size of messages delivered to the client may exceed the configured threshold.
When you set the dispatch rate limit in bytes/throttling-period (`dispatchThrottlingRateInByte`/`ratePeriodInSecond`), the broker calculates `the number of entries to read from bookies` in one throttling period through the following equation:
`The number of entries to read from bookies` = `The **total** byte size to read`
`The average byte size per entry`

By controlling `the number of entries to read from bookies`, the broker attempts to limit `the total byte size to read` below `the dispatch rate` within each throttling period. It reads messages from the bookies in the unit of `entry` and approximates the bytes of the next entry to read because it does not know the exact byte size of each entry before reading it.

The broker uses the following two metrics to get `the average byte size per entry`:

- Average publish size (`brk_ml_EntrySizeBuckets`): the average byte size per entry stored in the bookies when the broker receives a publish request.
- Average dispatch size (`entriesReadSize`/`entriesReadCount`): the average byte size per entry read from bookies, that is, the average byte size per entry sent to the client.

The broker uses the average publish size in preference to the average dispatch size. If the average publish size is unavailable, then it uses the average dispatch size. When none of the two metrics are available, the broker only reads one entry at the first attempt.

2) The number of messages delivered to the client may exceed the configured threshold.
When you set the dispatch rate limit in message-count/throttling-period (`dispatchThrottlingRateInMsg`/`ratePeriodInSecond`) and batching (`batch-send`) is enabled, the broker counts an entry as one message (despite the message count per entry) and calculates `the number of entries to read from bookies` through the following equation:

`The number of entries to read from bookies` = `The **total** number of messages to read`
`The average message count per entry` (=1)
Since there is a number of messages per entry, the number of messages delivered to the client always exceeds or equals the configured threshold.

Workaround
Configuring `preciseDispatcherFlowControl` or `dispatchThrottlingOnBatchMessageEnabled` can mitigate the over-delivery issue. For example, turning on `preciseDispatcherFlowControl` can mitigate the limitation by pre-decrementing the quota using the approximated average message count per entry.
See Throttling configurations for more details.

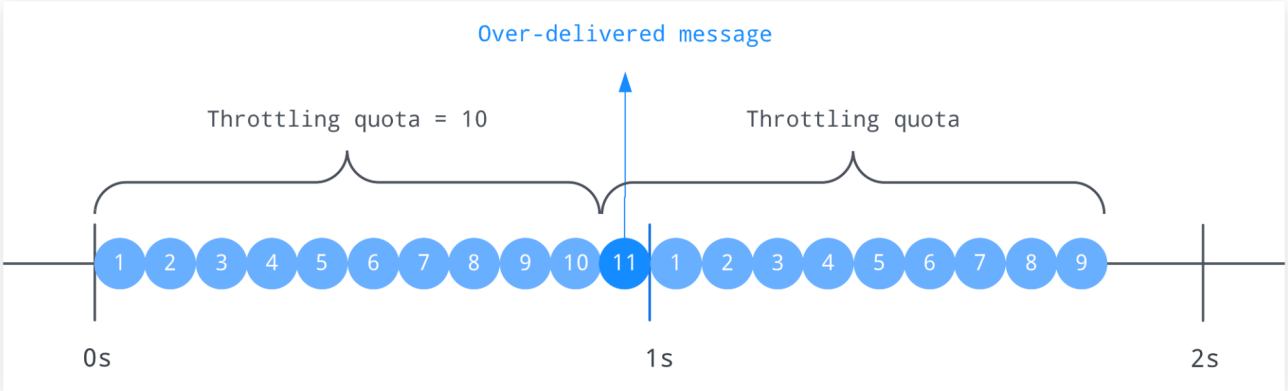
2. Concurrent throttling processes may not decrease the quota in a timely manner.
As introduced in How it works, the dispatch throttling process is: `1.get remaining quota` -> `2.load data` -> `3.decrease quota`. If two processes "dispatch replay messages (`process-R`)" and "dispatch non-replay messages(`process-N`)" in the same subscription are executed in parallel, their throttling processes can be interwoven in this order:

1) `process-R: 1.get remaining quota`
2) `process-R: 2.load data`
3) `process-N: 1.get remaining quota`
4) `process-N: 2.load data`
5) `process-R: 3.decrease quota`
6) `process-N: 3.decrease quota`

As a result, the total number of dispatched messages may exceed the quota.

Note: When over-delivery happens, and the delivered message count exceeds the quota in the current period, then the quota for the next period will be reduced accordingly. For example, if the rate limit is set to `10/s`, and `11` messages have been delivered to the client in the first period, then only up to `9` messages can be delivered to the client in the next period; if 30 messages have been delivered in the last period, the count of messages to deliver in the next two periods is `0`.

Original version: https://github.com/apache/pulsar/pull/19012.
New PR: https://github.com/apache/pulsar-site/pull/386.



Concepts

Throttling levels

The following table outlines the three levels that you can throttle message dispatch.

Level	Description
Per broker	All subscriptions in a single broker share the quota.
Per topic	All subscriptions in the same topic share the quota. - If it's a non-partitioned topic, the quota equals the maximum number of messages the topic can deliver per unit of time. - If a topic has multiple partitions, the quota refers to the maximum number of messages each partition can deliver per unit of time. In other words, the actual dispatch rate limit of a partitioned topic is N times the configured one (N is the number of partitions inside the topic). For example, the topic `t0` has two partitions. If you set the quota to `10/s`, then the rate limit of both `t0-p0` and `t0-p1` is `10/s`, while the total rate limit of `t0` is `20/s`. Note that the quota cannot be shared among partitions, but can be shared among subscriptions inside a partition.
Per subscription	- If it's a non-partitioned topic, the rate limit refers to the maximum number of messages that a subscription can deliver to clients per unit of time. - If the subscribed topic has multiple partitions, the rate limit refers to the maximum number of messages that the subscription can deliver per partition per unit of time. In other words, a subscription's actual dispatch rate limit for a partitioned topic is N times the configured one (N is the number of partitions inside the topic). For example, the topic `t1` has two partitions with the subscription `s1`. If you set the rate limit to `10/s`, then the rate limit for `s1` in both `t1-p0` and `t1-p1` is `10/s`, while the total rate limit of `s1` is `20/s`.

Note: The dispatch rate limits configured at multiple levels take effect simultaneously (logical AND).

Throttling approaches

The following table outlines multiple approaches to configure the dispatch rate limits at different levels.

Approach	Per cluster	Per topic	Per subscription
Set broker configs or dynamic broker configs.	• `dispatchThrottlingRateInMsg` • `dispatchThrottlingRateInByte` See Throttling configurations for more details.	• `dispatchThrottlingRatePerTopicInMsg` • `dispatchThrottlingRatePerTopicInByte` It applies to all topics in the cluster.	• `dispatchThrottlingRatePerSubscriptionInMsg` • `dispatchThrottlingRatePerSubscriptionInByte` It applies to all subscriptions in the cluster.
Set namespace policies	N/A	Refer to Configure dispatch throttling for topics. It applies to all topics in a namespace.	Refer to Configure dispatch throttling for subscriptions. It applies to all subscriptions in a namespace.
Set topic policies	N/A	Refer to Set topic-level dispatch rate.	Refer to Set subscription-level dispatch rate. It applies to all subscriptions in a topic.

Note: The dispatch rate limits configured through the above three approaches take effect with priorities, which is "topic policies" > "namespace policies" > "broker configurations". For example, if you have configured the dispatch rate limit for a subscription using all these three approaches, only the one configured through “topic policies” takes effect.

Throttling configurations

The following table outlines the parameters that you can configure for message dispatch throttling in the `conf/broker.conf` file.

Parameter	Description	Default value
`dispatchThrottlingRateInMsg`	The total number of messages that can be delivered per cluster per throttling period.	'-1', which means no limit.

Parameter	Description	Default value
	To set the topic-level or the subscription-level one, configure `dispatchThrottlingRatePerTopicInMsg` or `dispatchThrottlingRatePerSubscriptionInMsg`.	
`dispatchThrottlingRateInByte`	The total byte size of messages that can be delivered per cluster per throttling period. To set the topic-level or the subscription-level one, configure `dispatchThrottlingRatePerTopicInByte` or `dispatchThrottlingRatePerSubscriptionInByte`.	'-1', which means no limit.
`ratePeriodInSecond`	The period of time for dispatch throttling (in seconds). The counter is reset at the end of the period. For example, if you want to configure the rate limit to `10,000 messages per minute`, you need to set `ratePeriodInSecond` to `60` and set `dispatchThrottlingRateInMsg` to `10,000`.	1 (second)

`preciseDispatcherFlowControl`	Whether to apply a precise control on the dispatch throttling.
	By default, it's disabled, which means the broker approximates `the number of messages to read from bookies` using the minimum value between the remaining `consumer.receiverQueueSize` (defaults to 1000) and `dispatcherMaxReadBatchSize` (defaults to 100).
	
	When it's set to `true`, the broker approximates `the number of entries to read from bookies` through this equation:
	`the number of messages to read from bookies`
	`the average number of messages per entry`
	
	For example, assume you've set the rate limit to `10/s`. When the average number of messages per entry is `6`, it means the broker reads 2 entries per second, and the total number of messages to read exceeds the quota. However, when it's set to 'false`, the broker reads 10 entries (approximately 60 messages), which exceeds the quota much more.
		false

`dispatchThrottlingOnBatchMessageEnabled`	Whether to count messages by entry (batch). By default, it's disabled. Note that setting it to `true` may lead to an inaccurate approximation of the total message count but maximize Pulsar's throughput while keeping stable read requests to the bookies. For example, assume you've set the rate limit to `10/s`, if you set `dispatchThrottlingOnBatchMessageEnabled` to `true`, the broker	false

Parameter	Description	Default value
	only reads 10 entries and delivers them to the client per second, despite the number of messages per entry.	
`dispatchThrottlingOnNonBacklogConsumerEnabled`	Whether the dispatch throttling on non-backlog consumers is enabled. By default, it's enabled. When it is set to `false`: ● If all the consumers in one subscription have no backlog, the message dispatch throttling is turned off automatically even if `dispatchThrottlingRateInMsg` and `dispatchThrottlingRateInByte` are configured. ● If at least one consumer has a backlog, the throttling is turned on automatically.	true

Notes:

- You can use `dispatchThrottlingRateInMsg` and `dispatchThrottlingRateInByte` simultaneously (logical AND).
- Ensure that only one of `preciseDispatcherFlowControl` and `dispatchThrottlingOnBatchMessageEnabled` is enabled at one time since they are mutually exclusive. Both parameters can be used to improve the over-delivery issues (see Limitations). The difference between them is:
  - If `preciseDispatcherFlowControl` is enabled, Pulsar considers the number of messages per entry. This parameter takes effect when the broker reads entries from the bookies.
  - If `dispatchThrottlingOnBatchMessageEnabled` is enabled, Pulsar ignores the number of messages per entry. This parameter takes effect when the broker updates the counter after sending messages to the client.