

Cloudera Edge Management 2.2.0

Configuring EFM Bulk Operation Updates

Date published: 2019-04-15

Date modified: 2024-07-23

CLOUDERA

<https://docs.cloudera.com/>

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Fine-tuning bulk operation processing.....	4
Scenario: Frequency and size of batch operation.....	4
Scenario: Heartbeat interval and timeout rate.....	5
Scenario: Heartbeat interval for agent class.....	6
Scenario: Deployed state check frequency and timeout.....	7
Recommendations for bulk operations in EFM.....	8

Fine-tuning bulk operation processing

Learn how to configure Edge Flow Manager (EFM) bulk operation updates.

You can find the related configuration in the `efm.properties` file located in the `conf` directory of the EFM installation. EFM sends updates to agents through operations, which can be categorized into the following types:

- Agent level: Operations for individual agents, for example: debug requests
- Agent class level: Operations for all agents within an agent class, for example: flow updates, asset downloads, or property updates

This documentation describes various configuration scenarios to optimize agent class level updates.

To prevent heavy network loads and avoid Distributed Denial-of-Service (DDoS)-like behavior from agents, EFM sends updates to agents in smaller batches.

Below are the parameters related to operation updates:

```
# The maximum number of operations queued to be sent at once
efm.operation.monitoring.rollingBatchOperationsSize=100

# The number of operations to queue for different update types
efm.operation.monitoring.rollingOperationsSize.update.asset=10
efm.operation.monitoring.rollingOperationsSize.update.configuration=100
efm.operation.monitoring.rollingOperationsSize.update.properties=100
efm.operation.monitoring.rollingOperationsSize.sync.resource=100

# The interval for refreshing the operation queue
efm.operation.monitoring.rollingBatchOperationsFrequency=10s

# The multiplier for agent heartbeat interval properties to calculate timeout for queued operation
efm.operation.monitoring.inQueuedStateTimeoutHeartbeatRate=1.0

# The timeout for deployed, but not done operations
efm.operation.monitoring.inDeployedStateTimeout=5m

# The interval for checking operation timeouts
efm.operation.monitoring.inDeployedStateCheckFrequency=1m

# Enable/disable operation timeouts
efm.operation.monitoring.enabled=true

# The agent heartbeat interval monitor properties default value
efm.monitor.maxHeartbeatInterval=5m

# Class-specific values
efm.monitor.maxHeartbeatIntervalForClass.[Class\ A]=1m
efm.monitor.maxHeartbeatIntervalForClass.[Class\ B]=2m
efm.monitor.maxHeartbeatIntervalForClass.[Class\ C]=0m
```

Check out the different scenarios to see how changing these parameters can impact a deployment with 10,000 agents.

Scenario: Frequency and size of batch operation

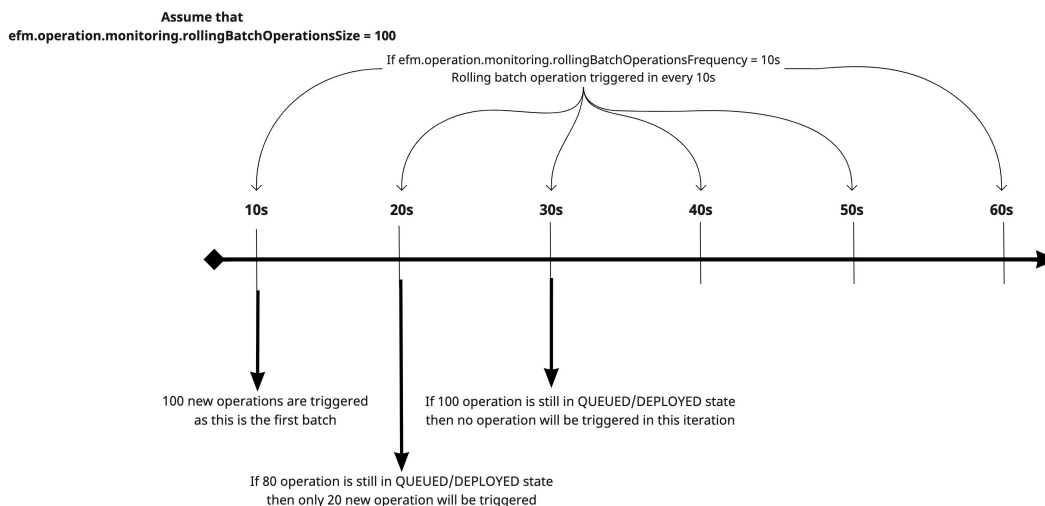
Learn how you can control the frequency of triggering new batch operations and the number of agents that should receive the new operation, and how you can resolve the issue of network saturation.

Scenario

You have 10,000 agents, and the operation is to download a 100 MB asset to each agent from EFM. The expectation is that an average agent is able to download an asset in 20-30 seconds (3-5MB/s). Assuming that EFM can handle 100MB/s network IO for this purpose, this means that roughly 20-30 parallel operations can be managed by EFM, which could easily lead to network saturation.

Analysis

In the following diagram, you can see that the `rollingBatchOperationsFrequency` parameter defines how often EFM triggers a new batch operation, and the `rollingBatchOperationsSize` parameter defines how many agents should receive the new operation.



Important: Increasing the value of the `rollingBatchOperationsFrequency` parameter slows down the trigger rate and mitigates load on EFM. However, setting it too high might not give agents enough time to finish with unnecessary triggers.



Important: Increasing the value of the `rollingBatchOperationsSize` parameter speeds up execution by allowing more agent operations at the same time. However, setting it too high can overload EFM.

Solution

A good starting point could be `rollingBatchOperationsSize=20` and `rollingBatchOperationsFrequency=5s`. In the first few iterations, the queue may become full, but it will stabilize, and the operations will proceed as expected. The expectation is that roughly 25% of the queue will be free in each iteration with a 5-second frequency.



Important:

- Executing this operation without the roll batch operation feature would cause 10K*4MB/s network load on EFM.
- Even with a slightly higher batch size (for example, 40), EFM can easily run out of new network I/O capacity and you would experience a DDoS-like load on EFM side.

Scenario: Heartbeat interval and timeout rate

Learn how you can control the interval between heartbeats and EFM operation timeouts, and resolve the issue of unhealthy agents.

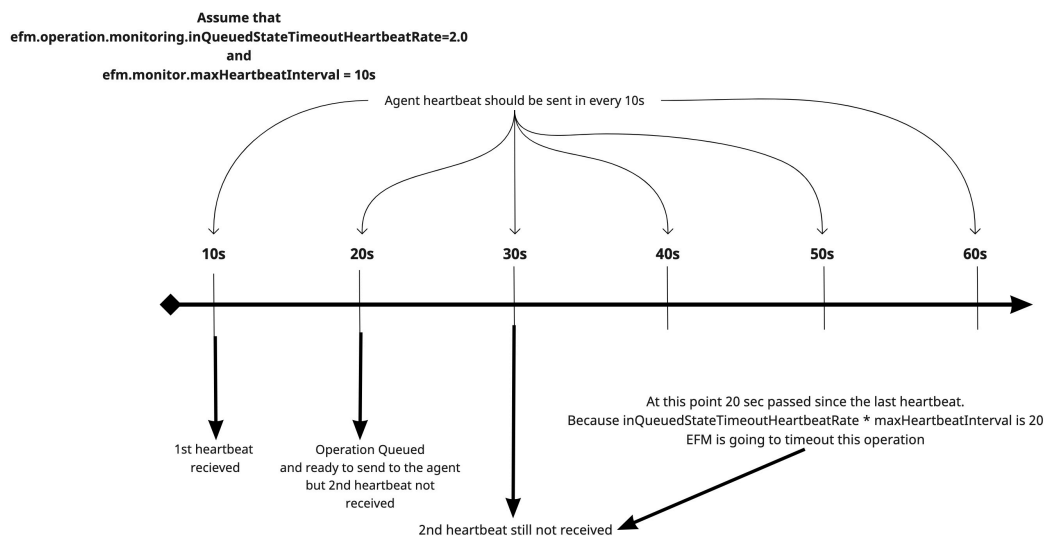
Scenario

You are running a bulk operation and some of your agents have unstable network access. When the router goes down, it takes one minute to recover. You know that agents are configured with 10-second heartbeat interval.

Analysis

When not all agents are healthy, you are faced with the issue that your bulk operation slows down drastically because of the lack of responses from agents. So the operation is stuck on the EFM side, unable to send updates to the agent. To manage this situation, you can define `efm.monitor.maxHeartbeatInterval=10s` and `efm.operation.monitoring.inQueuedStateTimeoutHeartbeatRate=2.0`.

The following chart shows you how this situation looks over time:



Important: Increasing the value of the `maxHeartbeatInterval` parameter allows more time for agents to respond. However, setting it too high can increase the total execution time.



Important: Increasing the value of the `inQueuedStateTimeoutHeartbeatRate` parameter allows more iteration for agents to come back online. However, setting it too high makes EFM wait unreasonably long for agents, allowing fewer new operations to be triggered during that time.

Solution

You can set `maxHeartbeatInterval=12` because if the agent is online, you must receive a heartbeat within 10 seconds, but this allows some buffer time. The 1-minute outage could cause you to miss 6-7 heartbeats. So you set `inQueuedStateTimeoutHeartbeatRate=8.`, meaning that EFM waits for 96 seconds before timing out the operation.



Important: Setting the timeout rate to 9 instead of 8 would cause ~10% longer execution time for agents where the operation times out.

Scenario: Heartbeat interval for agent class

Learn how you can control the the heartbeat interval at the agent class level and resolve related issues.

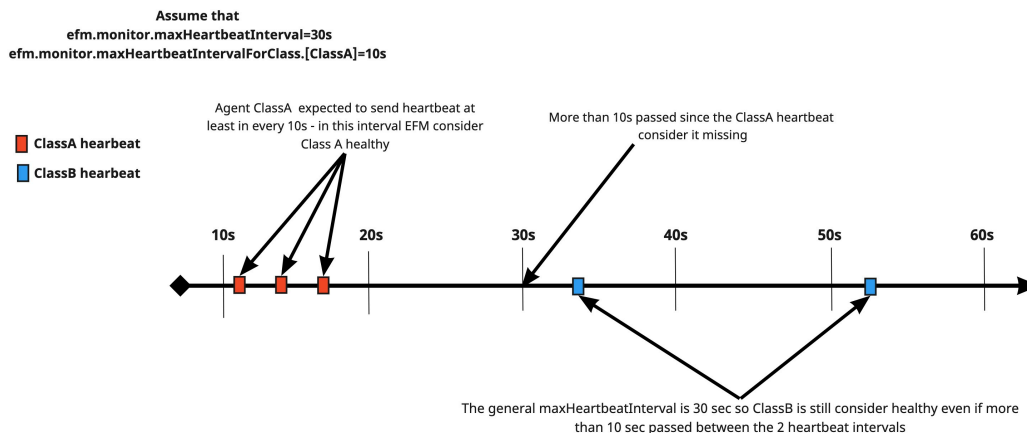
Scenario

You have an agent class with agents deployed in a new department where all the hardware and network performance exceed the rest of the system and heartbeat is configured to 5 seconds for those agents.

Analysis

You can configure the maximum heartbeat interval at the class level using the following parameter:
`efm.monitor.maxHeartbeatIntervalForClass.[ClassA]=10s`

The following chart shows how EFM considers an agent healthy based on these settings:



Important: Increasing the value of the `maxHeartbeatIntervalForClass` parameter, allows more time for agents in a specific class. However, too many unhealthy agents in this class would slow down bulk operations significantly.

Solution

You can introduce stricter expectations by setting `efm.monitor.maxHeartbeatIntervalForClass.[ClassA]=6s`. With `inQueuedStateTimeoutHeartbeatRate=8`, EFM waits only 48 seconds before timing out an operation and notifying you about the issue.

Scenario: Deployed state check frequency and timeout

Learn how you can control the frequency of checking the state of deployed operations and the timeout value for the deployed operations, and resolve related issues.

Scenario

A subset of agents have to manage other non-CEM applications that require a significant portion of network I/O, limiting the bandwidth available for the subset of agents compared to other agents. As a result, file download time is doubled compared to the expected 20-30 seconds.

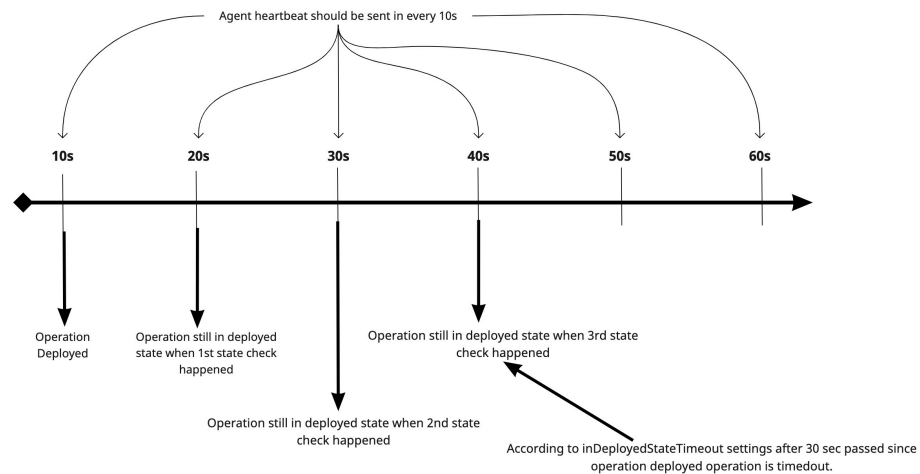
Analysis



Assume the agents are healthy, but the operation itself is time consuming. EFM CAN send an operation to multiple agents, but because the agents spend more time executing the operation, the operation remains in deployed state longer. To manage this situation, you can use the following parameters:

- `efm.operation.monitoring.inDeployedStateTimeout`
- `efm.operation.monitoring.inDeployedStateCheckFrequency`

The following chart illustrates how these parameters affect the behavior:

Assume that `efm.operation.monitoring.inDeployedStateTimeout=30s`
and
`efm.operation.monitoring.inDeployedStateCheckFrequency=10s`



-  **Important:** Increasing the value of the `inDeployedStateTimeout` parameter allows more time to execute an operation. However, if the operation is already completed, EFM might still not check the state, wasting execution time.
-  **Important:** Increasing the value of the `inDeployedStateCheckFrequency` parameter enables EFM to check the state more frequently, reducing wasted time. However, too frequent checks can burden the system during long-running deployments.

Solution

You can set `inDeployedStateTimeout=70` and `inDeployedStateCheckFrequency=5s`, which allows sufficient time for long running file downloads and ensures that EFM checks the state frequently enough to avoid wasting too much time.

Recommendations for bulk operations in EFM

Learn about guidelines for configuring Edge Flow Manager (EFM) properties to optimize bulk operations.

Cloudera recommends the following EFM property configurations:

- `efm.operation.monitoring.rollingBatchOperationsSize`: Set to 10-20% of the total number of agents, but ensure it does not exceed 1000.
- `efm.operation.monitoring.rollingOperationsSize.update.asset`,
`efm.operation.monitoring.rollingOperationsSize.update.configuration`,
`efm.operation.monitoring.rollingOperationsSize.update.properties`,
`efm.operation.monitoring.rollingOperationsSize.sync.resource`:

You can fine-tune the maximum number of simultaneous operations for each operation type. Ensure the values are aligned with `efm.operation.monitoring.rollingBatchOperationsSize`. If you increase the batch size, adjust the values of these properties accordingly. The total number of simultaneous operations will not exceed the value set in `efm.operation.monitoring.rollingBatchOperationsSize`, even if higher values are defined for the individual operation type properties.

For properties `efm.operation.monitoring.rollingOperationsSize.update.asset` and `efm.operation.monitoring.rollingOperationsSize.sync.resource`, the optimal value depends on the size of files being transferred.

- For small files (kilobytes magnitude), you can increase the limit similarly to the settings for `efm.operation.monitoring.rollingOperationsSize.update.properties`.

- For larger files (megabyte magnitude), keep the limit low, preferably not exceeding 10, to avoid performance issues.
- `efm.operation.monitoring.rollingBatchOperationsFrequency`: Based on former execution times, find the frequency where at most 25% of the rolling batch size frees up in a single iteration.
- `efm.monitor.maxHeartbeatInterval` in combination with `efm.operation.monitoring.inQueuedStateTimeoutHeartbeatRate`: Maxheartbeatrate should be close to 75 percentile so you can keep `inQueuedStateTimeoutHeartbeatRate` to a value which should not be more than 3. If a higher number is needed for the rate, you should investigate those agents why those agents do not match with criterias.
- `efm.operation.monitoring.inDeployedStateTimeout` in combination with `efm.operation.monitoring.inDeployedStateCheckFrequency`: Deployed state timeout should be 120% of the longest expected operation execution time. State check frequency should be set to such a value where EFM checks state at most 4-10 times during the operation execution.

If you combine all the configurations explained in the scenarios, you can derive an expected execution time formula:

$$\text{Expected Execution Time} = \overbrace{\frac{\text{numberOfOnlineAgents}}{\text{rollingBatchOperationsSize}} * \text{heartbeatPeriod}}^{\text{Healthy Agents impact on execution time}} + \overbrace{\frac{\text{numberOfMissingAgents}}{\text{rollingBatchOperationsSize}} * \text{inQueuedStateTimeoutHeartbeatRate} * \text{maxHeartbeatInterval[ForClass]}}^{\text{Unhealthy Agents impact on execution time}}$$