How to Measure Exalate Performance - the Ping Pong Test

Last Modified on 01/20/2026 6:13 am EST

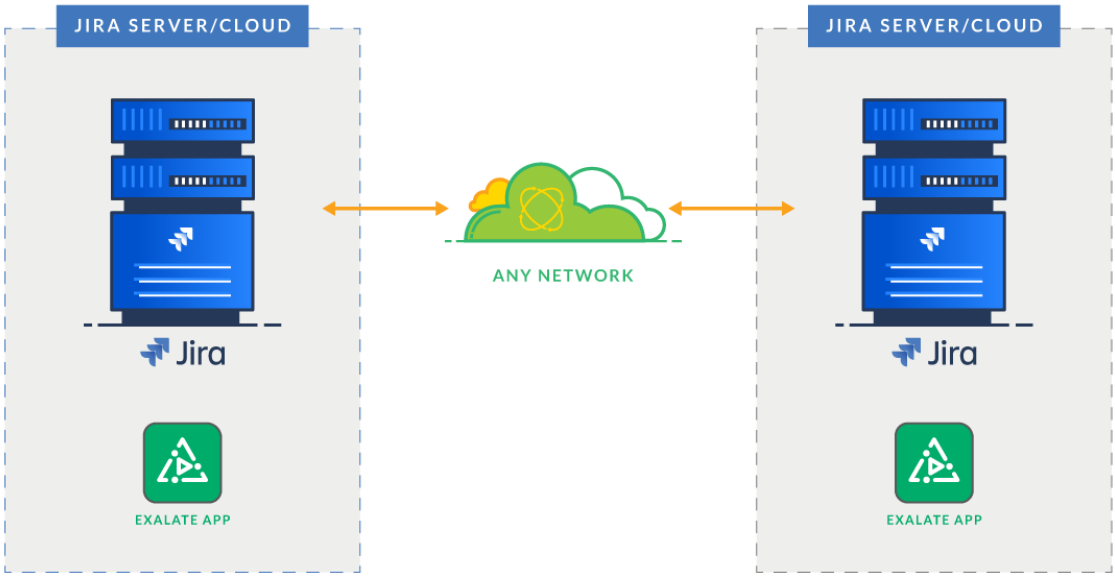
Introduction

The performance of Exalate depends on many different factors as it depends on many different components:

- the underlying trackers exalate is integrating with
- The machines hosting the Exalate instance (when deployed outside of the tracker)
- The network layout between the two different environments and the quality of the network connection
- The type and size of information that is being exchanged
- The complexity of the mapping and transformation of the synchronization

The ping pong test has been set up to have a benchmark such that any performance regression can be highlighted as these occur. This test also acts as a load test to check how the solution behaves under load.

Environment Setup



The left Jira has 2 projects

- Ping Pong Source (PPS)
- Ping Pong Target (PPT)

The right Jira has 1 project

- Ping Pong Wall (PPW)

The source has a set of 1000 issues, containing a mix of comments and attachments of various sizes. The Jira Data Generator add-on (here) can be used to create such projects.

The Ping Pong test

The ping pong test will validate:

- The Exalate operation (which brings an issue 'under sync')
- The sync back operation (which triggers a message back)
- The trigger operation (which automatically Exalates an issue)
- The Update operation (by updating the description this change needs to be applied to the target)
- The Unexalate operation (which severe the synchronization tie between two issues)

The Flow of a Single Issue

The issue keys and project keys are different in the actual test

	Description	Effect on PPS (left)	Effect on PPW (right)	Effect on PPT (left)	Exalate function
1	PPS-1 gets exalated using the ping_pong connection	-	-	-	Exalate
2	This creates an issue PPW-2 (on right jira)	-	create PPW-2		Create issue
3	The trigger on right Jira picks up the create event of PPW-2	-	-	-	synclistener captures create event
4	PPW-2 gets exalated using the ping_pong_part2 connection	-	-	-	trigger on PPW executes an exalate

	Description	Effect on PPS (left)	Effect on PPW (right)	Effect on PPT (left)	Exalate function
5	This creates on issue PPT-3 (on the left jira)			create PPT-2	create issue
6	The ping_pong_part2 has a syncback, an update syncevent is scheduled	-	-	-	Syncback is scheduling an update event
7	The incoming sync on PPW-2 updates a custom field 'Remote Key'		Field 'Remote Key' is updated with 'PPT-2'		issue is updated properly
8	The update is triggering a syncevent on the ping pong connection	-	-	-	synclistener captures update event
9	The ping_pong connection updates the custom field 'Remote Key'	Field 'Remote Key' is updated with 'PPT-2'	-	-	the issue is updated properly

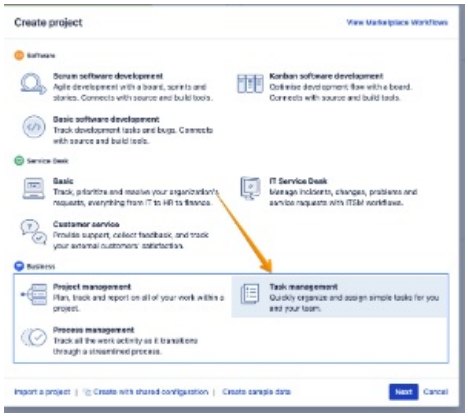
There are in total 9 exalate operations performed for one cycle.

Setting Up the Test

To configure the test, you will need to setup the following:

- Jira A and Jira B
  - Both on-premise
  - Both have Exalate deployed
- The projects
  - **PPS (Source - Jira A - Project Management configuration)**

    Project type



- PPW (Wall - Jira B - Project Management configuration)
- PPT (Target - Jira A - Project Management configuration)
- Additionally - on every project a custom field 'Remote Key' of type 'single line text'

- The Ping Connection
  - **Jira A**

    Jira A - Ping Connection - Outgoing sync

    The log.info is to collect the timestamps.

```
import java.sql.Timestamp

replica.key           = issue.key
replica.type          = issue.type
replica.assignee      = issue.assignee
replica.reporter      = issue.reporter
replica.summary       = issue.summary
replica.description   = issue.description
replica.labels        = issue.labels
replica.comments      = issue.comments
replica.resolution    = issue.resolution
replica.status        = issue.status
replica.parentId      = issue.parentId
replica.priority      = issue.priority
replica.attachments   = issue.attachments
replica.project       = issue.project

//Comment these lines out if you are interested in sending the full list of versions and components of the source project.
replica.project.versions = []
replica.project.components = []

log.info("PINGPONG - PING OUT - ${issue.key} - [${new Date().time}]")

/*
Custom Fields

replica.customFields."CF Name" = issue.customFields."CF Name"
*/
```

    Jira A - Ping Connection - Incoming sync

```
if(firstSync){
  // do not create on the outgoing path
  return
}


log.info("PINGPONG - PING IN  - ${issue.key} - [${new Date().time}]")
issue.summary      = replica.summary
issue.description  = replica.description
issue.labels       = replica.labels
issue.comments     = commentHelper.mergeComments(issue, replica)
issue.attachments  = attachmentHelper.mergeAttachments(issue, replica)
issue.customFields."Remote Key".value = replica.customKeys.pongissue
```

- **Jira B**

Jira B - Ping Connection - Outgoing Sync

```
replica.key            = issue.key
replica.type           = issue.type
replica.assignee       = issue.assignee
replica.reporter       = issue.reporter
replica.summary        = issue.summary
replica.description    = issue.description
replica.labels         = issue.labels
replica.comments       = issue.comments
replica.resolution     = issue.resolution
replica.status         = issue.status
replica.parentId       = issue.parentId
replica.priority       = issue.priority
replica.attachments    = issue.attachments
replica.project        = issue.project
replica.customKeys.pongissue = issue.customFields."Remote Key".value

//Comment these lines out if you are interested in sending the full list of versions and components of the source project.
replica.project.versions = []
replica.project.components = []
```

Jira B - Ping Connection - Outgoing sync

```
if(firstSync){
  issue.projectKey   = "PPW"
  issue.typeName     =  "Task"
}
issue.summary      = replica.summary
issue.description  = replica.description
issue.labels       = replica.labels
issue.comments     = commentHelper.mergeComments(issue, replica)
issue.attachments  = attachmentHelper.mergeAttachments(issue, replica)
```

- The Pong Connection

- **Jira A**

Jira A - Pong Connection - Outgoing Sync

```
replica.key            = issue.key
replica.type           = issue.type
replica.assignee       = issue.assignee
replica.reporter       = issue.reporter
replica.summary        = issue.summary
replica.description    = issue.description
replica.labels         = issue.labels
replica.comments       = issue.comments
replica.resolution     = issue.resolution
replica.status         = issue.status
replica.parentId       = issue.parentId
replica.priority       = issue.priority
replica.attachments    = issue.attachments
replica.project        = issue.project

//Comment these lines out if you are interested in sending the full list of versions and components of the source project.
replica.project.versions = []
replica.project.components = []


//replica.customKeys.foo = new Date()

/*
Custom Fields

replica.customFields."CF Name" = issue.customFields."CF Name"
*/
```

Jira A - Pong Connection - Incoming Sync

```
if(firstSync){
  issue.projectKey   = "PPT"
  // Set type name from source issue, if not found set a default
  issue.typeName     = "Task"

  // report back the issue key of the created issue
  syncHelper.syncBackAfterProcessing()
}
issue.summary      = replica.summary
issue.description  = replica.description
issue.labels       = replica.labels
issue.comments     = commentHelper.mergeComments(issue, replica)
issue.attachments  = attachmentHelper.mergeAttachments(issue, replica)
```

- **Jira B**

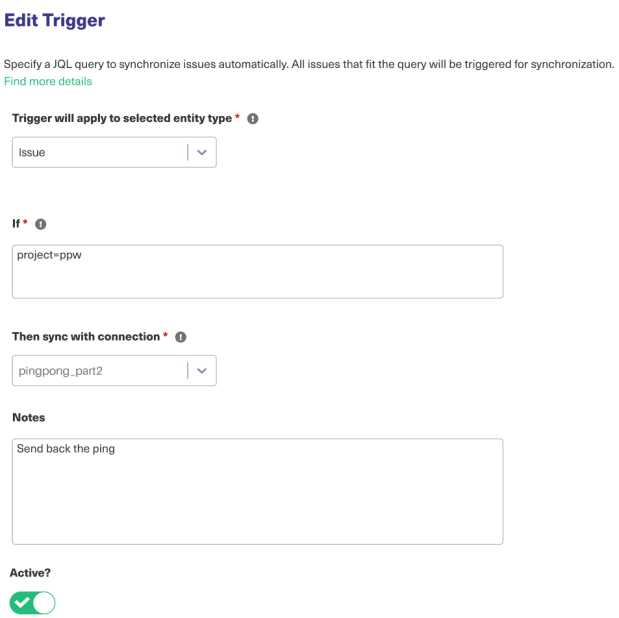
Jira B - Pong connection - Outgoing sync

```
replica.key          = issue.key
replica.type         = issue.type
replica.assignee     = issue.assignee
replica.reporter     = issue.reporter
replica.summary      = issue.summary
replica.description  = issue.description
replica.labels       = issue.labels
replica.comments     = issue.comments
replica.resolution   = issue.resolution
replica.status       = issue.status
replica.parentId     = issue.parentId
replica.priority     = issue.priority
replica.attachments  = issue.attachments
replica.project      = issue.project

//Comment these lines out if you are interested in sending the full list of versions and components of the source project.
replica.project.versions = []
replica.project.components = []
```

Jira B - Pong Connection - Incoming sync

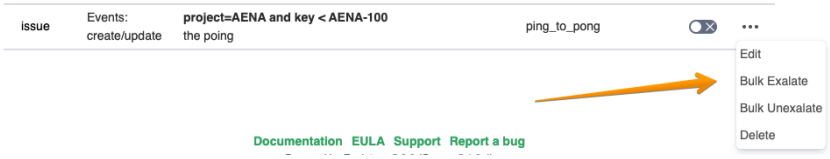
```
issue.summary     = replica.summary
issue.description = replica.description
issue.labels      = replica.labels
issue.comments    = commentHelper.mergeComments(issue, replica)
issue.attachments = attachmentHelper.mergeAttachments(issue, replica)


// the update of the custom field will trigger an update event on the ping connection back to source
issue.customFields."Remote Key".value = replica.key
```

- An active trigger that Exalates issues over the pong connection which are created on the PPW project



Running the Test

- Start an exalate on a subset of issues on project PPS by creating a trigger (with a JQL) and choosing Bulk Exalate.



- Inspect the logging (exalate.log in the <jira-home>/logs directory).
  Grep on the string 'PINGPONG' - it will reveal the timestamps.

What can you expect?

- As stated in the introduction, there are many components at play that will influence the outcome of the performance test.
- Our baseline, used in the regression tests, is to process on average 300 issues in an hour (2700 synchronization transactions)