Infrastructure - Feature #8766

support server-side link checking for the 303 redirect url in the resolve call

2019-02-15 19:03 - Rob Nahf

Status:	In Progress	**Start date:**	2019-02-15
Priority:	Normal	**Due date:**	
Assignee:	Rob Nahf	**% Done:**	30%
Category:	d1_cn_service	**Estimated time:**	0.00 hour
Target version:			
Milestone:	None	**Story Points:**	
Product Version:	*		

Description

cn.resolve returns a 303 "see other" redirect that browsers use instead of consuming the ObjectLocationList in the response payload. Occasionally, the link provided returns a "not found", and it could be either from a member node being temporarily down (so not listed as down in the node list), or not having the object requested (an invalid replica that hasn't been audited yet, so is still listed as 'completed').

There isn't a good way to send the client urls to all of the possible replicas (http code 300 "multiple choices", although defined, as a code, doesn't have standard way to handle responses, so isn't a real option), so the solution would have to be server-side.

Server-side solutions would increase the complexity, response time, and load on the CN, so any solution would have to take all these factors into account. If a reasonable solution exists for the first two constraints, perhaps the load issue could be solved by dedicating a server for resolve - or maybe it could run on one of the centralized replica MNs.

In slack convo I brainstormed a couple of possible ways:

```
rob   [20 hours ago]
hi @jing,  @peter,  Is LTER down temporarily or permanently?
cn.resolve will skip member nodes marked “down” in the nodelist, so
I can set it to a down status if it’s permanently offline, and the
redirect url will then point to the CN.  I don’t know how to
rearrange the order of replicas in the replica list.

rob   [20 hours ago]
Resolve was not designed to test the replicas before choosing one
for the redirect url.  The problem with checking is that down nodes
usually take a while for the call to timeout, so it has potential
performance impacts.

What might work, since listNodes is cached on the apache server, is
to ping all nodes for up/down status before returning the nodelist.
Because of the server-side caching, it wouldn’t constantly be
pinging the MNs, just every 3-5 minutes or so.

peter   [19 hours ago]
@rob LTER is up  https://gmn.lternet.edu/mn/v2/node. Isn’t it the
responsibility of the client to go through the list resolve returns?
I would say that it is also the responsibility of the client to
check the replication status as well, and find a replica that is
valid.

rob   [15 hours ago]
@peter, you’re right about it being the client responsibility to go
through the list, but the redirect kind of creates the impression of
something being wrong when the redirect fails, since that client
(browser) doesn’t consume the object location list, but follows the
redirect.

rob   [15 hours ago]
resolve does narrow the possibilities, though, by only returning
```

```
COMPLETED replicas from “up” nodes.

peter   [2 hours ago]
@rob do you mean that because `resolve` sets the HTTP Location to
the URL of the first location in the response, which could be down?
I think you mentioned previously that it would be hugely inefficient
for the CN to check every location. Not sure how this could be
improved.
See https://releases.dataone.org/online/api-documentation-v2.0/apis/CN_APIs.html#CNRead.resolve (e
dited)

rob   [2 hours ago]
yes, exactly.

rob   [1 hour ago]
I think there might be ways to minimize the inefficiency of the
extra lookups - shortening the http timeouts to 5 seconds and
performing the calls concurrently.  There’s undoubtedly a way to
continue when one of the threads returns with a live url so you
don’t have to wait in case one of the nodes being called is slow.

rob   [1 hour ago]
Keeping track of the responsiveness of the node would also help to
minimize the chance of calling a temporarily down node in the first
place.
```

I wrote some strawman code on the multi-threaded idea, attached for future use if we decide to move forward. It is reasonably easy to follow (low complexity), and seems to get around the slowness issue (performance), just need to test it for load it might add.

Related issues:

Related to Infrastructure - Bug #8812: Resolve service returns 500 for HTTP H...	**Closed**	**2019-05-22**

History

#1 - 2019-02-15 19:49 - Rob Nahf

- File UrlChecker.java added

- File ConfirmedNode.java added

Used together, these could be added to the cn.resolve implementation. They should be refined, performance tested, and especially load tested before being put in service.

The basic concept is that of a scavenger hunt: first (thread) to bring back a live link wins and the game is over. (Don't need to wait for the slower responders). Details in the comment in the classes, and example workflow in the main method (which can be removed when implemented).

#2 - 2019-05-22 19:59 - Rob Nahf

- Related to Bug #8812: Resolve service returns 500 for HTTP HEAD request added

#3 - 2019-05-31 15:07 - Rob Nahf

- Description updated

#4 - 2019-05-31 15:16 - Rob Nahf

- Description updated

Files

ConfirmedNode.java	1.02 KB	2019-02-15	Rob Nahf