Item 94
git_comments:

1. * Should be called holding the dlock

git_commits:

1. **summary:** Revert "GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r… (#3853)"
   message: Revert "GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r… (#3853)" This reverts commit 181e3a4a

github_issues:

github_issues_comments:

github_pulls:

1. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
   body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
2. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
   body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
3. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
   body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
4. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…

body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.

5. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
   body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
6. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
   body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
   label: test
7. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
   body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
   label: code-design
8. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
   body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this

PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.

9. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
   body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
10. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
   body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
11. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
   body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
12. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
   body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.

13. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
    body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
14. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
    body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
15. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
    body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
16. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
    body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
17. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
    body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit

within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
label: test

18. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
19. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
20. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
21. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.

22. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
    body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
23. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
    body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
24. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
    body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
25. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
    body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
    label: code-design
26. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
    body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this

PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
label: code-design

27. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
28. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
label: test
29. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
30. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please

ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
31. **title:** GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r…
body: …egion in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou Evans <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, check Concourse for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.

github_pulls_comments:

github_pulls_reviews:

1. mark this as @VisibleForTesting
2. move this typeToId.put before the synchronized since it has nothing to do with classToType. typeToId is a synchronized map.
3. This looks like a bug in some existing code that you are now using for buildTypeToIdFromRegion. This check should be moved in to the spot when we increment totalPdxTypeIdInDS. Or be changed to >=. We have the same pattern of code in: getExistingIdForEnum
4. I think you need to make the same changes to this class in how it handles "enums". See defineEnum and enumToId.
5. **body:** Add a unit test for PeerTypeRegistration and tests in it for the methods you are changing.
label: test
6. **body:** Since this builds both the typeToId and enumToId maps from the region it seems like you should name it so (like buildTypeAndEn umToIdMapsFromRegion). We should be able to get rid of getExistingIdForEnum but need to verify the max stuff on with two different counters in this method.
label: code-design
7. this method is called on afterCreate on a CacheListener. I think this listener should also update the enumToId map but it currently does not.
8. I purposely leave the enums untouched. It will be handled in another ticket.
9. done
10. done
11. done
12. enum will be fixed in another ticket.
13. What is the ticket number?
14. What is the ticket for the enum work?
15. I think this would also be part of the enum ticket
16. **body:** The while class has no junit tests. We always use dunit/integration test to cover. And this fix is also covered by dunit test. We will add one more integration test. According to Darrel's suggestion, I filed a new ticket GEODE-7034 to refactor the class to be junit testable.
label: test
17. GEODE-7035
18. GEODE-7035
19. GEODE-7035
20. Having "region" name in the function is not looking right; it will be nice to add what region, in this context. E.g.: idToTypeRegion how about: buildTypeToIdFromIdToType?
21. The method needs to be changed; as we are updating typeToId map.
22. done
23. done
24. **body:** we probably can add a performance benchmark and remove this test
label: code-design

25. **body:** Just a preference: I don't think these asserts are necessary. We would get a NPE instead of an assertion error.
    label: code-design
26. Possibly break this down into smaller sets of tests. Prove when using the system property for sorted JSON field objects, we only produce 1 type When there are collisions to the same calculated id, we do not lose a type
27. **body:** Possibly break up this test into smaller tests
    label: test
28. possibly remove this assertion
29. possibly extract product hashing code into a method and make it package protected so we use the actual product calculation

jira_issues:

1. **summary:** getExistingIdForType should not compare all entries in idToType region
   description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".
2. **summary:** getExistingIdForType should not compare all entries in idToType region
   description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit

test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".

3. **summary:** getExistingIdForType should not compare all entries in idToType region
   description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".
4. **summary:** getExistingIdForType should not compare all entries in idToType region
   description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode

conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".

5. **summary:** getExistingIdForType should not compare all entries in idToType region
   description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".
   label: code-design
6. **summary:** getExistingIdForType should not compare all entries in idToType region
   description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new

pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".
label: code-design

7. **summary:** getExistingIdForType should not compare all entries in idToType region
description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".
8. **summary:** getExistingIdForType should not compare all entries in idToType region
description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType

exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".

9. **summary:** getExistingIdForType should not compare all entries in idToType region
   description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".
10. **summary:** getExistingIdForType should not compare all entries in idToType region
    description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new

pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".

11. **summary:** getExistingIdForType should not compare all entries in idToType region
    description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".
12. **summary:** getExistingIdForType should not compare all entries in idToType region
    description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".
13. **summary:** getExistingIdForType should not compare all entries in idToType region

description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".

14. **summary:** getExistingIdForType should not compare all entries in idToType region
    description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".
15. **summary:** getExistingIdForType should not compare all entries in idToType region
    description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when

there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".

16. **summary:** getExistingIdForType should not compare all entries in idToType region
    description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".
17. **summary:** getExistingIdForType should not compare all entries in idToType region
    description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K

x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".

18. **summary:** getExistingIdForType should not compare all entries in idToType region
    description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".
19. **summary:** getExistingIdForType should not compare all entries in idToType region
    description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId

map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".

20. **summary:** getExistingIdForType should not compare all entries in idToType region
    description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".
21. **summary:** getExistingIdForType should not compare all entries in idToType region
    description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region

(which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".
label: code-design

22. **summary:** getExistingIdForType should not compare all entries in idToType region
    description: We found the PeerTypeRegistration's getExistingIdForType() will iterate through the idToType region's entries to find if the incoming newType is there. If idToType region contains 20K or 100K entries, this will impact the put throughput (customers did notice the performance downgrade when there're many pdxTypes). To make the things worse, the comparison is to compare the whole object, field to field. If the json object (which will be converted to pdxType) contains 30 fields, the comparison will have to compare up to 30 fields. If the idToType region contains 20K entries, A new pdxType will do 20K x 30 string comparisons before register it. We found each server maintained a typeToId map, this map is used to check if the pdxType exists. If exists, it will return the type id without check the IdToType region. The total number of pdxType did not impact the put performance if the pdxTypd exists. The typeToId map is maintained with a d-lock, each time we added a new pdxType, it will update into the map while still holding the d-lok. So we believe that the map should be the same as the region in content. If we cannot find the pdxType in the map, it should not be in the region. We can skip the iteration of region (which is the root cause of the performance issue). Another issue in current code is: when each time a new type come, it will recreate the map. This is unnecessary and contributes to the slowness too. We should only create the map during initialize(). Here are the tests we want to introduce: 1) a junit test to prove that reorder fields in a big JSON file will not cause significant hashcode conflicts (<1%) 2) a junit test to prove that add a index to a field in a big JSON file will hardly cause hashcode conflicts. This 2 tests are to prove that hashcode conflict is not the root cause of linear probing for PDXTypeId. 3) a junit test to prove that for the cases that hashcode conflict caused by reordered fields, there will be no hashcode conflicts if using SORT_JSON_FIELD_NAMES_PROPERTY=true. 4) a dunit test to prove that SORT_JSON_FIELD_NAMES_PROPERTY=true or false did not impact the performance to add a new pdxType. 5) a dunit test to create a new pdxType from 2 peer server at the same time. The test is to prove that the d-lock take effect, one server create the pdxType, and another server should find the pdxType exists. Do this test both from server directly and from clients. 6) Create 2 different objects which ends up with the same hashcode (we can get the 2 objects from test-1), try to put the 2 objects to create new pdxType. The 2nd one should also create a new type. It should not be treated as "found an existing pdxType".

jira_issues_comments:

1. Commit 181e3a4a465aa9f5e06f39cd1285c94f9bc78600 in geode's branch refs/heads/develop from Xiaojian Zhou [https://gitbox.apache.org/repos/asf?p=geode.git;h=181e3a4] GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r… (#3853) * GEODE-6973: Use cachelistener to synchronize typeToId with IdToType region in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io>
2. Commit 181e3a4a465aa9f5e06f39cd1285c94f9bc78600 in geode's branch refs/heads/develop from Xiaojian Zhou [https://gitbox.apache.org/repos/asf?p=geode.git;h=181e3a4] GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r… (#3853) * GEODE-6973: Use cachelistener to synchronize typeToId with IdToType region in PeerTypeRegistrationm, then when creating a new json pdxType, no need to look up in IdToType region. This will speed up creating new pdxType. Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io>

3. Commit 2047c40d59ea31967bcbc36ae7c62e6489cdb74e in geode's branch refs/heads/develop from Naburun Nag [https://gitbox.apache.org/repos/asf?p=geode.git;h=2047c40] Revert "GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r… (#3853)" This reverts commit 181e3a4a
4. Commit 10677330666d6bbc45ac535268db026d1351872e in geode's branch refs/heads/release/1.10.0 from Naburun Nag [https://gitbox.apache.org/repos/asf?p=geode.git;h=1067733] Revert "GEODE-6973: Use cachelistener to synchronize typeToId with IdToType r… (#3853)" This reverts commit 181e3a4a465aa9f5e06f39cd1285c94f9bc78600.
5. **body:** Commit cdca78889a052f78f44dac945ea3f5b2ca32305e in geode's branch refs/heads/develop from Xiaojian Zhou [https://gitbox.apache.org/repos/asf?p=geode.git;h=cdca788] GEODE-6973: Use cachelistener to synchronize typeToId with IdToType (#4014) * GEODE-6973: Use cachelistener to synchronize typeToId with IdToType Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> * introduce a tmpTypeToId map for listener * Fixed some test failures Co-authored-by: Donal Evans <doevans@pivotal.io> Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> * fix junit test * enumId should also be put into the tmp map when listener is triggered * consolidate some comments of reviewers * change the name of the local map * add back load into map in initalize, since it will be fast. * Added DUnit test Co-authored-by: Donal Evans <doevans@pivotal.io> Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> * Spotless apply Authored-by: Donal Evans <doevans@pivotal.io> * Refactoring DUnit tests Authored-by: Donal Evans <doevans@pivotal.io> * Removing bad tests Authored-by: Donal Evans <doevans@pivotal.io> * refactor the local maps * Corrected reload from region qualifier Authored-by: Donal Evans <doevans@pivotal.io> * Use method to access PdxTypes region Authored-by: Donal Evans <doevans@pivotal.io> * save() should have a boolean parameter to determine if to save into pending map
   label: code-design
6. **body:** Commit cdca78889a052f78f44dac945ea3f5b2ca32305e in geode's branch refs/heads/develop from Xiaojian Zhou [https://gitbox.apache.org/repos/asf?p=geode.git;h=cdca788] GEODE-6973: Use cachelistener to synchronize typeToId with IdToType (#4014) * GEODE-6973: Use cachelistener to synchronize typeToId with IdToType Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io> * introduce a tmpTypeToId map for listener * Fixed some test failures Co-authored-by: Donal Evans <doevans@pivotal.io> Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> * fix junit test * enumId should also be put into the tmp map when listener is triggered * consolidate some comments of reviewers * change the name of the local map * add back load into map in initalize, since it will be fast. * Added DUnit test Co-authored-by: Donal Evans <doevans@pivotal.io> Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> * Spotless apply Authored-by: Donal Evans <doevans@pivotal.io> * Refactoring DUnit tests Authored-by: Donal Evans <doevans@pivotal.io> * Removing bad tests Authored-by: Donal Evans <doevans@pivotal.io> * refactor the local maps * Corrected reload from region qualifier Authored-by: Donal Evans <doevans@pivotal.io> * Use method to access PdxTypes region Authored-by: Donal Evans <doevans@pivotal.io> * save() should have a boolean parameter to determine if to save into pending map
   label: code-design
7. Commit 6bc9249cfdc3ff48cdb6c36b83d7e1b22a1a952c in geode's branch refs/heads/revert-4014-feature/GEODE-6973-2 from Xiaojian Zhou [https://gitbox.apache.org/repos/asf?p=geode.git;h=6bc9249] Revert "GEODE-6973: Use cachelistener to synchronize typeToId with IdToType (#4014)" This reverts commit cdca78889a052f78f44dac945ea3f5b2ca32305e.
8. Commit 4b9148e6f477e495995737f345f74376238d43e5 in geode's branch refs/heads/develop from zhouxh [https://gitbox.apache.org/repos/asf?p=geode.git;h=4b9148e] Revert "GEODE-6973: Use cachelistener to synchronize typeToId with IdToType (#4014)" This reverts commit cdca78889a052f78f44dac945ea3f5b2ca32305e. Click too fast and the message is messy. Will re-merge.
9. Commit 18ccf564e1bb34dba891c75681474f572a35fcbf in geode's branch refs/heads/feature/GEODE-6973 from zhouxh [https://gitbox.apache.org/repos/asf?p=geode.git;h=18ccf56] GEODE-6973: Use cachelistener to synchronize typeToId with IdToType Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io>
10. Commit 18ccf564e1bb34dba891c75681474f572a35fcbf in geode's branch refs/heads/develop from zhouxh [https://gitbox.apache.org/repos/asf?p=geode.git;h=18ccf56] GEODE-6973: Use cachelistener to synchronize typeToId with IdToType Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io>
11. Commit 3b834f28b66787a156c0f69cb010611326f81f21 in geode's branch refs/heads/feature/GEODE-6973 from zhouxh [https://gitbox.apache.org/repos/asf?p=geode.git;h=3b834f2] GEODE-6973: The pdxRegion.size() should be called outside of TX context to avoid messaging. Co-authored-by: Anil

<agingade@pivotal.io> Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io>

12. Commit f3d6fe42d1f668131b1d7d6b45a94438b3e9001e in geode's branch refs/heads/feature/GEODE-6973 from zhouxh [https://gitbox.apache.org/repos/asf?p=geode.git;h=f3d6fe4] GEODE-6973: The pdxRegion.size() should be called outside of TX context to avoid messaging. Co-authored-by: Anil <agingade@pivotal.io> Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io>
13. Commit 5874bfc19ff22b09c44a1a1ee4fb641627ff8def in geode's branch refs/heads/develop from Xiaojian Zhou [https://gitbox.apache.org/repos/asf?p=geode.git;h=5874bfc] GEODE-6973: The pdxRegion.size() should be called outside of TX context to avoid messaging. (#4233) Co-authored-by: Anil <agingade@pivotal.io> Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io>
14. Commit 5874bfc19ff22b09c44a1a1ee4fb641627ff8def in geode's branch refs/heads/feature/GEODE-7341 from Xiaojian Zhou [https://gitbox.apache.org/repos/asf?p=geode.git;h=5874bfc] GEODE-6973: The pdxRegion.size() should be called outside of TX context to avoid messaging. (#4233) Co-authored-by: Anil <agingade@pivotal.io> Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io>
15. Commit 5874bfc19ff22b09c44a1a1ee4fb641627ff8def in geode's branch refs/heads/feature/GEODE-7341 from Xiaojian Zhou [https://gitbox.apache.org/repos/asf?p=geode.git;h=5874bfc] GEODE-6973: The pdxRegion.size() should be called outside of TX context to avoid messaging. (#4233) Co-authored-by: Anil <agingade@pivotal.io> Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io>
16. Commit 5874bfc19ff22b09c44a1a1ee4fb641627ff8def in geode's branch refs/heads/feature/GEODE-7341 from Xiaojian Zhou [https://gitbox.apache.org/repos/asf?p=geode.git;h=5874bfc] GEODE-6973: The pdxRegion.size() should be called outside of TX context to avoid messaging. (#4233) Co-authored-by: Anil <agingade@pivotal.io> Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io>
17. Commit 5874bfc19ff22b09c44a1a1ee4fb641627ff8def in geode's branch refs/heads/feature/GEODE-7341 from Xiaojian Zhou [https://gitbox.apache.org/repos/asf?p=geode.git;h=5874bfc] GEODE-6973: The pdxRegion.size() should be called outside of TX context to avoid messaging. (#4233) Co-authored-by: Anil <agingade@pivotal.io> Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io>
18. Commit 5874bfc19ff22b09c44a1a1ee4fb641627ff8def in geode's branch refs/heads/feature/GEODE-7341 from Xiaojian Zhou [https://gitbox.apache.org/repos/asf?p=geode.git;h=5874bfc] GEODE-6973: The pdxRegion.size() should be called outside of TX context to avoid messaging. (#4233) Co-authored-by: Anil <agingade@pivotal.io> Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io>
19. Commit 5874bfc19ff22b09c44a1a1ee4fb641627ff8def in geode's branch refs/heads/feature/GEODE-7258 from Xiaojian Zhou [https://gitbox.apache.org/repos/asf?p=geode.git;h=5874bfc] GEODE-6973: The pdxRegion.size() should be called outside of TX context to avoid messaging. (#4233) Co-authored-by: Anil <agingade@pivotal.io> Co-authored-by: Xiaojian Zhou <gzhou@pivotal.io> Co-authored-by: Donal Evans <doevans@pivotal.io>
20. Transition from Resolved to Closed for Apache Geode 1.11.0 RC4 release.