Item 336
git_comments:

1. write access synced
2. * * To be called only by the owner of this object's monitor lock

git_commits:

1. **summary:** LUCENE-9453 Assert lock held before volatile write (#1734)
   message: LUCENE-9453 Assert lock held before volatile write (#1734) Found via IntelliJ warnings.

github_issues:

github_issues_comments:

github_pulls:

1. **title:** LUCENE-9453 Add sync around volatile write
   body: checkoutAndBlock is not synchronized, but has a non-atomic write to numPending. Meanwhile, all of the other writes to numPending are in sync methods. In this case it turns out to be ok because all of the code paths calling this method are already sync: `synchronized doAfterDocument -> checkout -> checkoutAndBlock` `checkoutLargestNonPendingWriter -> synchronized(this) -> checkout -> checkoutAndBlock` If we make synchronized checkoutAndBlock that protects us against future changes, shouldn't cause any performance impact since the code paths will already be going through a sync block, and will make an IntelliJ warning go away. Found via IntelliJ warnings.
   https://issues.apache.org/jira/browse/LUCENE-9453

github_pulls_comments:

1. this lock is left out intentionally to reduce the overhead of acquiring a reentrant lock. We only call this method internally from a method that guarantees the lock is held. I don't think we should add this, for correctness it's not necessary at this point. Adding something like `assert Thread.holdsLock(this);` I'd be ok with.

github_pulls_reviews:

jira_issues:

1. **summary:** DocumentWriterFlushControl missing explicit sync on write
   description: checkoutAndBlock is not synchronized, but has a non-atomic write to {{numPending}}. Meanwhile, all of the other writes to numPending are in sync methods. In this case it turns out to be ok because all of the code paths calling this method are already sync: {{synchronized doAfterDocument -> checkout -> checkoutAndBlock}} {{checkoutLargestNonPendingWriter -> synchronized(this) -> checkout -> checkoutAndBlock}} If we make {{synchronized checkoutAndBlock}} that protects us against future changes, shouldn't cause any performance impact since the code paths will already be going through a sync block, and will make an IntelliJ warning go away.

jira_issues_comments:

1. I believe it might have been left out intentionally. A reentrant lock is fairly cheap but still adds overhead. If this method is private and called from under monitors already then this would be sufficient to ensure correctness (and eliminate any extra monitor overhead): assert Thread.holdsLock(this) : "This method should never be called without a lock!";
2. Commit 092076ec39e0f71ae92d36cd4ebe69e21a97ce4e in lucene-solr's branch refs/heads/master from Mike Drob [https://gitbox.apache.org/repos/asf?p=lucene-solr.git;h=092076e] LUCENE-9453 Assert lock held before volatile write (#1734) Found via IntelliJ warnings.
3. Thanks for the feedback [~dweiss], [~simonw]. Added the assert and committed this.