http://mail-archives.apache.org/mod\_mbox/lucene-dev/200110.mbox/%3C3BC4B124.8060306@earthlink.net%3E

From dmitrys@earthlink.net Wed Oct 10 20:36:01 2001

Return-Path: <dmitrys@earthlink.net>

Mailing-List: contact lucene-dev-help@jakarta.apache.org; run by ezmlm

Delivered-To: mailing list lucene-dev@jakarta.apache.org

Received: (qmail 58462 invoked from network); 10 Oct 2001 20:36:01 -0000

Received: from relay4.uswest.net (HELO relay2.uswest.net) (63.226.138.12)

by daedalus.apache.org with SMTP; 10 Oct 2001 20:36:01 -0000

Received: (qmail 6523 invoked by uid 0); 10 Oct 2001 20:36:06 -0000

Received: from unknown (HELO earthlink.net) (65.100.117.194)

by relay2.uswest.net with SMTP; 10 Oct 2001 20:36:06 -0000

Message-ID: <3BC4B124.8060306@earthlink.net>

Date: Wed, 10 Oct 2001 14:35:48 -0600

From: Dmitry Serebrennikov <dmitrys@earthlink.net>

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:0.9.2) Gecko/20010726 Netscape6/6.1

X-Accept-Language: en-us

MIME-Version: 1.0

To: lucene-dev@jakarta.apache.org

Subject: multithreading in SegmentsReader

Content-Type: text/plain; charset=us-ascii; format=flowed

Content-Transfer-Encoding: 7bit

X-Spam-Rating: daedalus.apache.org 1.6.2 0/1000/N

This fixes a potential race condition in SegmentsReader where a call to

numDocs() could return -1 if a document is being deleted at the same

time by another thread.

(Caviat: I don't actually have a test case that shows this problem, nor

was I able to verify this fix. Found it by reading the code and it

seemed easy enough and important enough to put in a fix.)

Also, I think there may be a larger problem with this class that happens

when close() is called concurrently with any of the methods that iterate

over the sub-readers and delegate to them. It looks like the second

thread may get an IOException if it calls on a reader that has been

closed. Not sure how to fix this without killing performance. Should we

just document this as a "feature" and say that its application's

responsibility to ensure that all queries are finished before calling

close()?

===================================================================

RCS file:

/home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/SegmentsReader.java,v

retrieving revision 1.1.1.1

diff -w -u -r1.1.1.1 SegmentsReader.java

--- SegmentsReader.java 2001/09/18 16:29:54 1.1.1.1

+++ SegmentsReader.java 2001/10/10 20:10:10

@@ -78,7 +78,18 @@

}

public final int numDocs() {

- if (numDocs == -1) { // check cache

+ // Synchro: two goals -- prevent duplicate cache revalidation (minor)

+ // and prevent invalidation of the cache between checking it and

returning

+ // the value.

+

+ // Atomically copy it. Even if it changes between the

+ // if and the return, we still return a number that was valid

+ // when we entered this method.

+ final int tmpNumDocs = numDocs;

+ if (tmpNumDocs != -1) return tmpNumDocs; // check cache

+

+ synchronized(this) {

+ if (numDocs == -1) {

int n = 0; // cache miss--recompute

for (int i = 0; i < readers.length; i++)

n += readers[i].numDocs(); // sum from readers

@@ -86,6 +97,7 @@

}

return numDocs;

}

+ }

public final int maxDoc() {

return maxDoc;

@@ -102,10 +114,13 @@

}

public final void delete(int n) throws IOException {

+ // Synchro: synchronize with the cache re-validation logic in numDocs

+ synchronized(this) {

numDocs = -1; // invalidate cache

int i = readerIndex(n); // find segment num

readers[i].delete(n - starts[i]); // dispatch to segment

reader

}

+ }