Arvados - Bug #6194

[SDK] Support writing more than 64MiB at a time in Python SDK

05/29/2015 08:13 PM - Peter Amstutz

Status:	Resolved	**Start date:**	06/01/2015
Priority:	Normal	**Due date:**	
Assigned To:	Peter Amstutz	**% Done:**	100%
Category:	SDKs	**Estimated time:**	0.00 hour
Target version:	2015-06-10 sprint		

Description

Subtasks:

Task # 6210: Review 6194-python-arvfile-large-write	**Resolved**

Related issues:

Related to Arvados - Story #3198: [FUSE] Writable streaming arv-mount	**Resolved**	**04/14/2015**

Associated revisions

Revision 4b3b3064 - 06/02/2015 08:23 PM - Peter Amstutz

Merge branch '6194-python-arvfile-large-write' closes #6194

History

#1 - 05/29/2015 08:25 PM - Peter Amstutz

- Target version changed from Bug Triage to 2015-06-10 sprint

#2 - 05/29/2015 08:26 PM - Peter Amstutz

- Project changed from Arvados Private to Arvados

- Category set to SDKs

- Status changed from New to In Progress

- Assigned To set to Peter Amstutz

#3 - 05/29/2015 08:27 PM - Peter Amstutz

- Subject changed from [SDK] SUpport writing more than 64MiB at a time in Python SDK to [SDK] Support writing more than 64MiB at a time in Python SDK

#4 - 06/01/2015 09:14 PM - Tom Clegg

c766dd2

It looks like the second condition will always be True:

```
            while (n + config.KEEP_BLOCK_SIZE) < len(data):
                self.writeto(offset+n, dataview[n:n + config.KEEP_BLOCK_SIZE].tobytes(), num_retries)
                n += config.KEEP_BLOCK_SIZE
            if n < len(data):
                self.writeto(offset+n, dataview[n:].tobytes(), num_retries)
            return
```

The last three lines could be just "return self.writeto(offset+n, ...)"

Or, for even less copy & paste, we could just do this, since dataview[n:n+config.KEEP_BLOCK_SIZE] already returns a short segment when appropriate:

```
            while n < len(data):
                self.writeto(offset+n, dataview[n:n + config.KEEP_BLOCK_SIZE].tobytes(), num_retries)
                n += config.KEEP_BLOCK_SIZE
            return
```

I think it's worthwhile to make the test a bit stronger by using a non-zero offset (if there were an error in offset math, this is the only test that would catch it).

#5 - 06/02/2015 05:42 PM - Peter Amstutz

Simplified the write loop. Thanks.

I changed the test to do two writes:

```
text = "0123456789"
writer.write(text)
text = "0123456789" * 9999999
writer.write(text)
```

This has an interesting side effect. The first write ends up committed in its own block, because the current logic in writeto() commits the buffer block and starts a new one when (buffered data + new data) is bigger than the block size.

```
. 781e5e245d69b566979b86e28d23f2c7+10 48dd23ea1645fd47d789804d71b5bb8e+67108864 77c57dc6ac5a10bb2205caaa731879
94+32891126 0:100000000:count.txt\n
```

Changing the behavior of writeto() to split differently should be out of scope for this fix, but I wanted to flag it in case you had thoughts. I should mention that the existing behavior is intentional, the reasoning being that when a record or line of text is written in a single write() call, this biases towards not splitting records or lines across block boundaries.

#6 - 06/02/2015 07:50 PM - Tom Clegg

Peter Amstutz wrote:

> I changed the test to do two writes:
>
> [...]

Hm. The new test supplies a non-zero offset, which is good, but it doesn't seem to *test* that the offset was actually used: the file is 100000000 bytes long when the test starts, so it will still be 100000000 bytes long if offset is ignored -- for that matter, even if no writing happens at all.

Using "----------" * 3 in the first write(), and asserting that the manifest has the right md5sums after doing the big write, might be a reasonable way to make the test effective...?

Meanwhile, most of the following assertions seem to be just copied from other tests and I'm not sure why they add any value here:

```
+            self.assertEqual(None, c.manifest_locator())
+            self.assertEqual(True, c.modified())
+            c.save_new("test_write_large")
+            self.assertEqual("zzzzz-4zz18-mockcollection0", c.manifest_locator())
+            self.assertEqual(False, c.modified())
```

(In particular, do we really need to check that the UUID hasn't changed? I suppose this isn't the right time to wonder why we're using a method called *_locator() to get a UUID...)

> This has an interesting side effect. The first write ends up committed in its own block, because the current logic in writeto() commits the buffer block and starts a new one when (buffered data + new data) is bigger than the block size.

Indeed, I'm pretty sure it would be better in the long run to batch the writes according to max block size irrespective of the way the client chunks them on the way into write(). For example, that way "cat <a >b" and "dd if=a of=b" and "egrep . <a >b" would result in the same chunking. But I agree that should go into #3198 or some other story.

#7 - 06/02/2015 08:12 PM - Tom Clegg

Sorry, cancel that... misread the MockApi as being the initial content.

The boilerplate assertions still look kind of superfluous, but that doesn't need to block the bugfix.

LGTM, thanks.

#8 - 06/02/2015 08:16 PM - Peter Amstutz

Tom Clegg wrote:

> Peter Amstutz wrote:
>
> > I changed the test to do two writes:
> >
> > [...]
>
> Hm. The new test supplies a non-zero offset, which is good, but it doesn't seem to *test* that the offset was actually used: the file is 100000000 bytes long when the test starts, so it will still be 100000000 bytes long if offset is ignored -- for that matter, even if no writing happens at all.

That's not true, the file is 0 length when the test starts. Added an assertion to that effect.

> Using "----------" * 3 in the first write(), and asserting that the manifest has the right md5sums after doing the big write, might be a reasonable way to make the test effective...?

Well, the thought was that test_write_large (which writes 1000 bytes 100000 times) and test_large_write (which writes of 1000 bytes and then writes 99999990 bytes) would produce the same manifest, then I realized that they don't because of the size of writes affects how blocks are packed.

> Meanwhile, most of the following assertions seem to be just copied from other tests and I'm not sure why they add any value here:
>
> [...]

Those assertions are not necessary to test what we're testing, so I simplified it and just assert the manifest_text() directly.

> Indeed, I'm pretty sure it would be better in the long run to batch the writes according to max block size irrespective of the way the client chunks them on the way into write(). For example, that way "cat <a >b" and "dd if=a of=b" and "egrep . <a >b" would result in the same chunking. But I agree that should go into #3198 or some other story.

I see the argument for consistency in how blocks are packed, independent of the size of writes. We can do that in #3198 (not here).

#9 - 06/02/2015 08:25 PM - Peter Amstutz

- Status changed from In Progress to Resolved

- % Done changed from 0 to 100

Applied in changeset arvados|commit:4b3b3064b87a07b2ba8035dd5c8f3660dd3b2a67.