Arvados - Bug #10813

Improve performance of arv-put

01/04/2017 08:20 PM - Tom Morris

Status:	Resolved	**Start date:**	01/04/2017
Priority:	Normal	**Due date:**	
Assigned To:	Lucas Di Pentima	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2017-02-01 sprint		

Description

Uploading BCL files using arv-put only achieves 5-10 MB/s while using 35% CPU. This is to slow on transfer and too high on CPU usage. It also appears that performance consistently drifts down over the course of an upload, indicating, perhaps, an issue with processing large manifests.

Here's a little ASCII art graphic from bmon:

```
     MiB                      (RX Bytes/hour)                                             MiB
                  (TX Bytes/hour)
   48.40 ....................................|||......................                   8.04 .....
.........................|||...........................
   40.33 ....................................|||......................                   6.70 .....
............||||||||||||||...........................
   32.26 ...................................||||......................                   5.36 .....
||||||||||||||||||||||||||.........|||||.............
   24.20 |..................................||||......................                   4.02 .....
||||||||||||||||||||||||||.........|||||.............
   16.13 |..................................||||......................                   2.68 ....|
|||||||||||||||||||||||||||......||||||.............
    8.07 |::::::::::::::::::::::::::::::::::||||::::::::.............                   1.34 ::::|
|||||||||||||||||||||||||||::::::||||||.............
         1   5   10   15   20   25   30   35   40   45   50   55   60                      1   5
   10   15   20   25   30   35   40   45   50   55   60
```

The 50 MB/s download in hours 39-36 is from Azure blob storage to a local shell node using the blobxfer utility. The arv-put bandwidth starts at ~8 MB/s in hour 33 and drifts down to ~5 MB/s in hour 6, averaging 6 MB/s for the entire 30 hours.

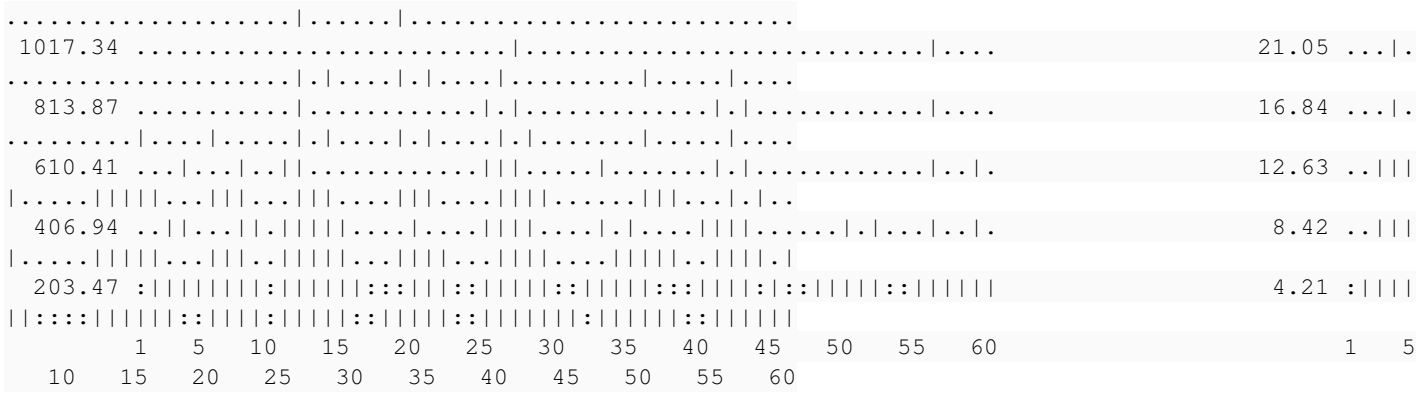
The performance goal is at least a 4x improvement to 25 MB/s, but achieving parity with blobxfer (~50 MB/s) would be even better.

Here's a detailed bandwidth graph of what a blobxfer transfer looks like:

```
    MiB                      (RX Bytes/second)                                           KiB
                (TX Bytes/second)
   97.84 ..|........................||....|..............|.......                  133.63 .....
....................|................................
   81.54 ..||....||...||....||...|||...||....||...||....||...|||...||                 111.36 .....
....................|..........||.....................
   65.23 ..|||..|||...|||..|||...|||..|||..|||...||...|||...|||...||                  89.09 ..||.
..|||...|||...||...|||...||....||....|....|....|||....|
   48.92 ..|||..|||...|||..|||...|||...|||..|||...|||..|||...|||..|||                  66.82 ..|||
..|||...|||..|||...|||...|||..|||...||...|||...|||...||
   32.61 ..|||..|||..||||..||||..|||..||||..|||..||||..||||..|||..|||                  44.54 ..|||
..|||...|||..||||..|||...|||..|||...|||..||||..|||..|||
   16.31 :|||||||||:||||::||||:|||||:||||::||||:|||||:||||::|||::|||                  22.27 ::|||
||||||:||||::|||||||||:||||::||||:|||||:||||::|||::|||
         1   5   10   15   20   25   30   35   40   45   50   55   60                      1   5
   10   15   20   25   30   35   40   45   50   55   60
```

Here's the corresponding arv-put graph

```
     KiB                     (RX Bytes/second)                                            MiB
                (TX Bytes/second)
 1220.81 ..................................................|....                   25.25 .....
```



All of the Illumina sequencer outputs are pretty similar: ~600 GB in ~242,000 files, the bulk of which are ~238,000 gzipped BCL files that range in size from 2 MB to 4 MB with the following size distribution:

```
 202477 3 MB
  33461 4 MB
   2141 2 MB
      1 1 MB
```

The files are grouped in directories of about 300 MB each, like this:

```
79058    Data/Intensities/BaseCalls/L005
310    Data/Intensities/BaseCalls/L005/C309.1
310    Data/Intensities/BaseCalls/L005/C308.1
```

The blobxfer utility uses 6 worker threads by default and it looks from the gaps in the bandwidth graph like that's not sufficient to cover all the latency with these small files sizes, but arv-put is doing much worse.

Subtasks:

Task # 11008: Review 10813-arv-put-six-threads	**Resolved**
Task # 10818: Review 10813-arv-put-six-threads	**Resolved**

Associated revisions

Revision c8aa6553 - 01/18/2017 09:32 PM - Lucas Di Pentima

Merge branch '10813-arv-put-six-threads'
Refs #10813

Revision 7a53cfc9 - 01/31/2017 03:13 PM - Lucas Di Pentima

Merge branch '10813-arv-put-six-threads'
Closes #10813

History

#1 - 01/04/2017 08:24 PM - Tom Morris

- Assigned To set to Lucas Di Pentima

#2 - 01/04/2017 08:38 PM - Tom Morris

- Story points set to 1.0

#3 - 01/04/2017 09:13 PM - Tom Morris

- Description updated

#4 - 01/04/2017 10:16 PM - Lucas Di Pentima

Tom, can you please describe also the dataset being uploaded? File count, and size distribution would help. Thanks!

#5 - 01/05/2017 04:06 AM - Tom Morris

- Description updated

#6 - 01/06/2017 06:54 PM - Lucas Di Pentima

- Status changed from New to In Progress

#7 - 01/09/2017 10:48 PM - Tom Morris

The other use case is upload of output collections from Crunch jobs to Keep. Although I'm sure the small files of BCLs don't help things, the problem exists with large files too. An output collection of 125 GB spread over just 16 files uploads at an average of less than 10 MB/s.

#8 - 01/13/2017 07:54 PM - Lucas Di Pentima

- File arv-put perf.ods added

Did some *back box* testing using arvagrant to test the performance with different network speeds. I've attached an OOo spreadsheet with the results and test setup. In short, arv-put performed between 80% and 100% of the control case (scp command), and on a special case it outperformed scp because the CPU started to be the bottleneck.

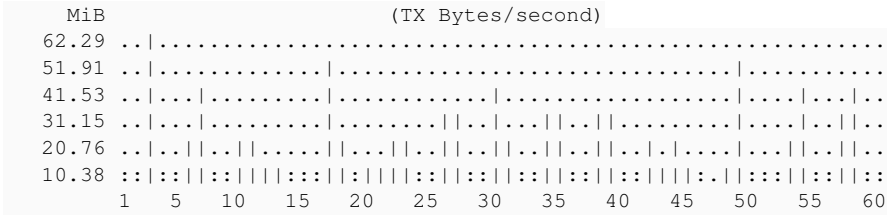
On arvagrant, when the network speed went to 1Gbps, arv-put occupied 60% of the bandwidth when uploading 2MB files, so I tried adding more upload threads but didn't got any significant enhancement. Nevertheless, the arvagrant test speeds have been a lot greater than what it was seen on production.

The next step was to try on an Azure installation, I tried c97qk with different thread numbers. It seems that the latency when using Azure blob storage is the problem, we can see spikes up to 60MB/s so there's enough bandwidth to reach high transfer rates but using the default 2 threads averages at a 40% of the capacity.

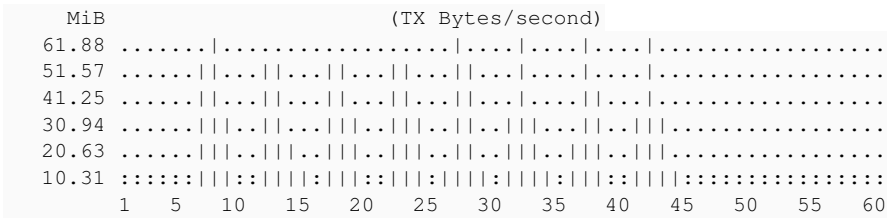
When adding threads, the network usage graphs get ever less spiked until we reach 6 threads, over that value there's no significant improvement.

Thread tests (Queue size = 1 / 1GB upload - 512 files 2MB each)

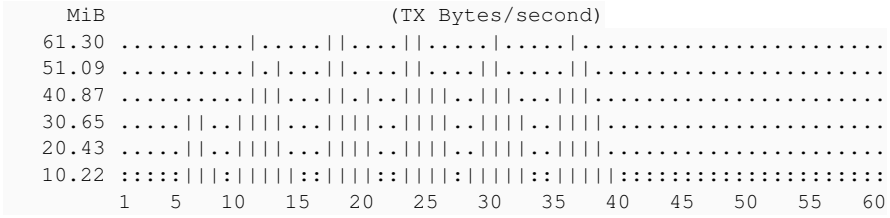
1 Thread: 73 secs ~14 MB/s



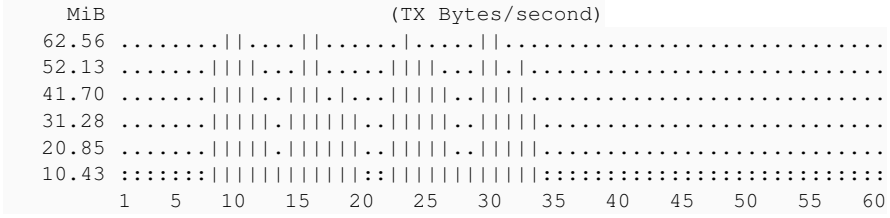
2 Threads: 41 secs ~25 MB/s



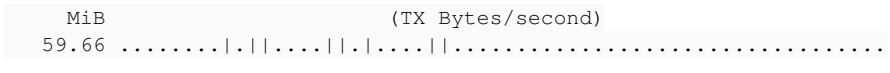
3 Threads: 37 secs ~28 MB/s



4 Threads: 30 secs ~34 MB/s



5 Threads: 28 secs ~36 MB/s



```
  49.71 ........|||||..||||....|||.|..............................
  39.77 ........|||||..|||||||.|||||..............................
  29.83 ........|||||||.|||||||||||||.............................
  19.89 ......|||||||||.|||||||||||||.............................
   9.94 ::::::|||||||||||||||||||||||||:::::.:::::::::::::..:::::::::
        1   5   10   15   20   25   30   35   40   45   50   55   60
```

6 Threads: 26 secs ~39 MB/s

```
    MiB                     (TX Bytes/second)
  59.74 ........|||...||......||.....................................
  49.78 ........|||..|||||....|||||..................................
  39.83 ........|||.|||||||||||||||||................................
  29.87 .......||||.|||||||||||||||||................................
  19.91 .......|||||||||||||||||||||||...............................
   9.96 :::::::|||||||||||||||||||||||||:::::::::::::.:::::::.:::::::::
        1   5   10   15   20   25   30   35   40   45   50   55   60
```

7 Threads: 26 secs ~39 MB/s

```
    MiB                     (TX Bytes/second)
  60.56 ......|||....|....|..........................................
  50.46 ......|||.|.|||||.||.||......................................
  40.37 ......|||||||||||||||||||....................................
  30.28 .....|||||||||||||||||||.....................................
  20.19 ....|||||||||||||||||||.......................................
  10.09 ::::|||||||||||||||||||||||||:::::::..:::::::::::::::::::::::
        1   5   10   15   20   25   30   35   40   45   50   55   60
```

8 Threads: 26 secs ~39 MB/s

```
    MiB                     (TX Bytes/second)
  62.03 ......|||..|...|.............................................
  51.69 ......|||.|||||||.||.||......................................
  41.35 .....||||||||||||.|||||||....................................
  31.01 .....|||||||||||||||||||.....................................
  20.68 .....|||||||||||||||||||.....................................
  10.34 :::|||||||||||||||||||||||:::::::::::::::::::::::::::::::::::
        1   5   10   15   20   25   30   35   40   45   50   55   60
```

#9 - 01/17/2017 05:07 PM - Tom Morris

Here's the results of a test run on an Azure production cluster. The starting bandwidth of ~32 MB/s is good, but it tails off to less then a quarter of that over the course of the upload.

```
    MiB                      (TX Bytes/hour)
  31.93 ..............||.....................................
  26.61 ..............||.....................................
  21.29 ............||||.....................................
  15.97 ..........||||||.....................................
  10.64 ....||.|.|||||||.....................................
   5.32 :::|||||||||||||:::::::::::::::::::::::::::::::::::::::::::
        1   5   10   15   20   25   30   35   40   45   50   55   60
```

I'm running a test with 8 threads now on the same 2 core shell node. We'll need to choose whether we go with a fixed default (4? 6? 8?) or scale it to the number of cores like blobxfer does (e.g. 2 * num_cpus)

#10 - 01/18/2017 06:28 AM - Tom Morris

I don't have final results, but 8 threads appears to be a little slower, if any anything, but not much different.

There is however a very distinctive and regular pattern to the CPU usage. After a couple of hours, it looks like:

I 8-9 seconds @ 50%
II 2-3 seconds @5-10%
III 11-12 seconds @100%

At the beginning of the arv-put job, Phase I was longer (~16-17 sec) and Phase III shorter (~2 sec).

My interpretation of this is that:
- Phase I is the data transfer phase
- Phase II represents a draining/sync period when less and less work is done until all threads are completely quiescent
- Phase III is the manifest save
- the 20 second manifest save cycle mentioned by Lucas is measured from beginning of save to beginning of save, so the longer and longer manifest

saves end up stealing more and more of the transfer time
- something(s) is/are braindead in the manifest save code causing it to take 10s of seconds to write kilobytes of data

Suggestions:
- make manifest save period minutes (3? 5?), not seconds. If the 12 hour job I'm running dies, I don't really care if I miss 10 seconds (or even 100s).
- fix the manifest code! it's slowing down everything in the system (but I'm happy to have that pushed to a separate ticket and declare this one done with the changes in default threads and manifest save period)

#11 - 01/18/2017 02:43 PM - Lucas Di Pentima

Thanks Tom for your suggestions.

Your were right about Phase II: when the update thread asked for the manifest with only committed blocks for state saving, there was a bug that only happened when uploading file collections with subdirectories inside, that turned off the "only_committed_blocks" feature and called for a complete block flush of all the put threads, so the entire upload process stopped for some seconds.

I've fixed that bug and now the manifest acquiring process is done in parallel with the upload threads as it should be from the beggining.
I added --threads N argument to arv-put command, so that it can be used to override the default number of put threads (that's 2 at the moment).

Updates on branch 10813-arv-put-six-threads at b0e6c00
Test running at: https://ci.curoverse.com/job/developer-run-tests/138/

#12 - 01/18/2017 03:46 PM - Peter Amstutz

put.py:142 "troughput" -> "throughput"

I suggest adding the only_committed flag to Collection.manifest_text() (if True, skip commit_all()) so that put.py isn't required to call the internal method _get_manifest_text().

#13 - 01/18/2017 04:33 PM - Lucas Di Pentima

Updates pushed at 221c7d2
Tests running on: https://ci.curoverse.com/job/developer-run-tests/139/

#14 - 01/18/2017 06:59 PM - Lucas Di Pentima

- Target version changed from 2017-01-18 sprint to 2017-02-01 sprint

#15 - 01/18/2017 08:15 PM - Lucas Di Pentima

Merged master into branch: c01ce0788adade520dc825152685aee0449a7da4
Running tests at: https://ci.curoverse.com/job/developer-run-tests/140/

#16 - 01/18/2017 09:18 PM - Peter Amstutz

put.py line 523, the "." argument is unnecessary:

```
manifest = self._local_collection.manifest_text(strip=False,
                                                 normalize=False,
                                                 only_committed=True)
```

arvfile.py line 1073: For consistency, this should accept the only_committed optional parameter (also needs to be fixed to use is_bufferblock() instead of testing the locator name directly):

```
    def manifest_text(self, stream_name=".", portable_locators=False, normalize=False):
```

#17 - 01/30/2017 09:32 PM - Lucas Di Pentima

Updates: e502060
Tests: https://ci.curoverse.com/job/developer-run-tests/150/

Addressed both comments and merged master into the branch.

#18 - 01/31/2017 03:07 PM - Peter Amstutz

Lucas Di Pentima wrote:

> Updates: e502060
> Tests: https://ci.curoverse.com/job/developer-run-tests/150/
>
> Addressed both comments and merged master into the branch.

LGTM

#19 - 01/31/2017 03:15 PM - Lucas Di Pentima

- Status changed from In Progress to Resolved

- % Done changed from 50 to 100

Applied in changeset arvados|commit:7a53cfc92d4bca452a687db0a6f338e6deb1564a.

#20 - 01/31/2017 10:03 PM - Tom Morris

Just to record some of the final performance figures, I did three 600+ GB uploads with the following results:

- 2 threads (default) - 19.2 GB/s
- 6 threads - 44.7 GB/s
- 6 threads - 44.9 GB/s

This compares to the starting performance of 5-6 GB/s, representing a ~7x improvement and shaving over a day (~26 hrs) off the upload times. The goals were "at least a 4x improvement to 25 MB/s, but achieving parity with blobxfer (~50 MB/s) would be even better," so I'd say we achieved the stretch goal.

Files

arv-put perf.ods	33.2 KB	01/13/2017	Lucas Di Pentima