SFO15-207: Storage and filesystem optimisations

Steve Capper
Yazen Ghannam
Agenda

1. Introduction
2. Where we are currently with:
   a. Ceph
   b. Linux kernel
   c. Hadoop FS (HDFS)
3. Future work
4. Q & A
Ceph

- Ceph is a distributed storage system written in cross-platform friendly C++.
- Ceph was introduced at the last Connect:
- It did “just work” on ARM, but there were a couple of areas we were able to improve for AArch64.
Ceph - CRC32c

- Example: rados bench write for 60s on btrfs
Ceph - CRC32c

- This was mentioned at last Connect, but as a quick recap...
- CRC32c (Castagnoli polynomial) code was accelerated using the AArch64 optional instructions.
- This is in Ceph v9.0.0.
Ceph - IO Analysis

- Having taken a look at the CPU cycle distribution with \texttt{perf}, we moved on to examine general IO.
- \texttt{sar} was used to get a general picture of things (disk, network, ...).
- We drilled into block IO via \texttt{blktrace}.
Ceph - Effect of \texttt{PAGE\_SIZE} on writes

Chart showing effect of \texttt{PAGE\_SIZE} on writes/second needed for RBD bench write

- \texttt{wr\_sec/s} 64KB
- \texttt{wr\_sec/s} 4KB

Time
Ceph - `PAGE_SIZE` effects on Journal

- The Journal assumed that `PAGE_SIZE` was a good value for the storage sector size.
- Thus for a system running with a 64KB `PAGE_SIZE` more IO was carried out.
- This has been fixed in upstream git commit:
  - 2eb096a FileJournal: Remove CEPH_PAGE_SIZE assumptions
Ceph - Patched FileJournal

Total (ceph.baseline.journal.out):
Reads Queued: 0, 0KiB  Writes Queued: 68,395, 2,336MiB
Read Dispatches: 0, 0KiB  Write Dispatches: 62,439, 2,302MiB
Reads Requeued: 0  Writes Requeued: 19,847
Reads Completed: 0, 0KiB  Writes Completed: 51,766, 2,302MiB
Read Merges: 0, 0KiB  Write Merges: 13,985, 449,690KiB
IO unplugs: 21,510  Timer unplugs: 0

Total (ceph.patch.journal.out):
Reads Queued: 0, 0KiB  Writes Queued: 82,133, 660,698KiB
Read Dispatches: 0, 0KiB  Write Dispatches: 84,269, 646,526KiB
Reads Requeued: 0  Writes Requeued: 27,868
Reads Completed: 0, 0KiB  Writes Completed: 65,163, 646,482KiB
Read Merges: 0, 0KiB  Write Merges: 15,467, 123,500KiB
IO unplugs: 18,502  Timer unplugs: 0
Ceph - PAGE_SIZE and Bufferlist

- I was advised to check out the Bufferlist.
- A 64KB PAGE_SIZE led to a ~ 10x increase in peak memory usage by the MetaData Server (MDS)!
- This has been fixed in:
  - 4524316 Common: Do not use CEPH_PAGE_SIZE when appending buffers in Ceph
Teuthology + Ceph-QA Suite

- We also wanted to ensure that Ceph worked on ARM. :-).
- The Ceph-QA suite is comprised of a LOT (~5500) tests.
- We spent time running through these tests.
- The Bufferlist patch in the previous slide was also found to help some of the tests pass.
Linux Kernel - CRC32

- Both CRC32 and CRC32c checksums were implemented in 3.19 using optional AArch64 instructions.
- They are used extensively by btrfs and the Ceph kernel mode client code (rbd and cephfs).
Linux Kernel: Testing kprobes & uprobes

- AArch64 kprobes are being worked on by David Long at Linaro.
- AArch64 uprobes by Pratyush Anand at Redhat. uprobes requires kprobes.
- We analysed Ceph’s memcpy size utilisation using early versions of uprobes:
uprobes + perf + flame graphs
Hadoop HDFS

- More on Hadoop in the Hadoop session.
- Optimisation wise, we have CRC32c coded up by Ed Nevill for the Hadoop native library.
  - [https://issues.apache.org/jira/browse/HADOOP-11660](https://issues.apache.org/jira/browse/HADOOP-11660)
- micro-benchmark speedups of ~11x found.
- This will make it into the Hadoop 2.8 release.
Ceph - CPU usage - Future work
Future Work

- We have kicked the tyres with Ceph, using dev boards.
- As higher spec AArch64 hardware becomes more readily available - one will be able to stress the CPU/Linux kernel even more and spot new areas of interest.
Future Work (2)

- Ceph utilises a collection of libraries, some notable ones:
  - tcmalloc (from the Google perf tools)
  - Boost
  - RocksDB

- These libraries could benefit from some analysis on AArch64.
Future Work (3)

- Investigate pipelined CRC32c implementation on Ceph and Linux Kernel.
- Hadoop is already using a pipelined implementation resulting in ~11x speedup vs ~4.5x speedup with single-issue implementation.
Thank you for your attention!

Any questions/comments?