Action in Multiple WAL

Introduction

After efforts of several JIRA, now HBase can fully support multiple write-ahead-log (WAL). Multiple WAL will not only increase the write throughput, but will also supply the ability to accelerate replication speed, enable namespace-level write isolation, make sync writes with low latency possible, etc. and we will talk about details in this document. First of all, we will introduce the design and how to use multiple wal. Then we will illustrate the performance data in both testing environment and online cluster. At last we will talk about what could be done in future work.
Work involved here includes HBASE-5699, HBASE-6617, HBASE-14306, HBASE-14441, HBASE-14448 and HBASE-14456.

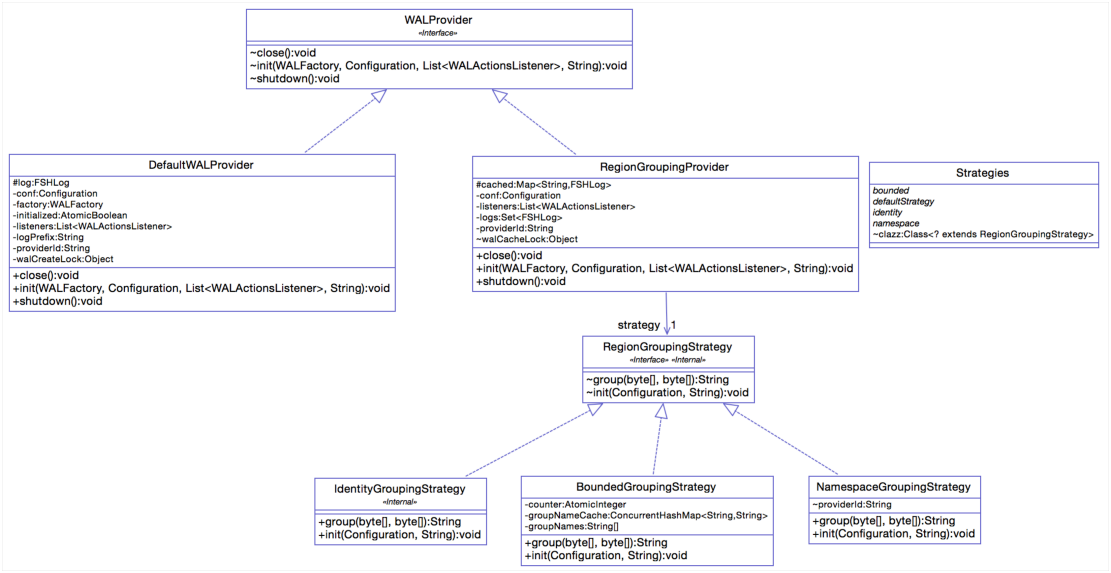
Implementation

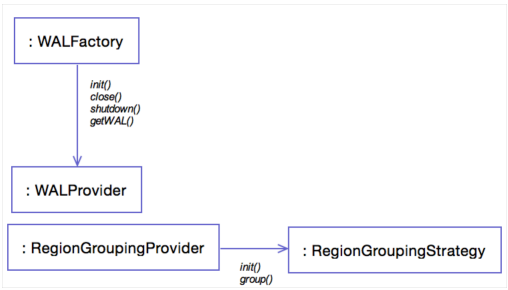
Excluding special wal for system tables such as meta and namespace, previously there will be only one WAL for each RegionServer, so the HRegionServer thread will directly instantiate FSHLog object and keep writing into it. However, after HBASE-5699, we have introduced a new WALProvider interface and delegate all WAL instantiation work to it. There're mainly two types of WALProvider: the DefaultWALProvider that each RegionServer remains to have one single WAL, and the new RegionGroupingProvider supporting multiple WAL. The WAL selection for writes is determined by the RegionGroupingStrategy. Currently we support 3 kinds of RegionGroupingStrategy:

- IdentityGroupingStrategy: one WAL for each region
  - ✓ Upside of this strategy is that flush triggered by count of WAL files reaching hbase.regionserver.maxlogs will be limited inside region level, downside is that in product environment this will introduce too many wal files
- BoundedGroupingStrategy: number of WALs is bounded, regions will be mapped to WAL following round-robin algorithm
  - ✓ This is the default strategy, and the default bound is 2.
- NamespaceGroupingStrategy: one WAL for each namespace
  - ✓ This enables namespace-level isolation: writes for one namespace won't affect those for another namespace anymore. This also makes it possible to monitor replication speed for each business, assuming each namespace representing one business

Below is the class and interaction diagram, for better understanding in

implementation perspective





To enable multiple WAL, add below settings in your hbase-site.xml:

```
<property>
  <name>hbase.wal.provider</name>
  <value>multiwal</value>
</property>
```

And if you'd like to specify the region grouping strategy, set below property

```
<property>
  <name>hbase.wal.regiongrouping.strategy</name>
  <value>bounded</value>
</property>
```

Available strategies are **identity**, **bounded** and **namespace**, bounded as the default. See strategies introduced above. You can also customize your own strategy and set the full-qualified class name here.

Notice that hbase.regionserver.maxlogs now limits each "regiongroup", so remember to adjust the number according to the groups you'll have with the chosen strategy. For example, if you are using the bounded grouping strategy and wal number is set to 4, setting hbase.regionserver.maxlogs to 32 would have the similar effect as previous setting 128

Evaluation

Test result with pure SATA disks

In this section we will show the testing result with PerformanceEvaluation tool.

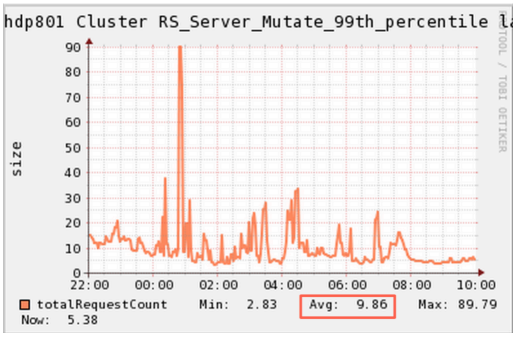
- Testing environment
  - 1 Master, 3 RegionServers, HDFS overlay
  - Hardware:
    - Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz
    - Network: 2*1GB, Intel Corporation I350 Gigabit Network Connection
    - MEM: 128G
    - Disk: 12*2TB, ST2000NM0033-9ZM175
  - Software:
    - HBase 1.1.2
    - HDFS 2.6.0
- Test command
  - hbase org.apache.hadoop.hbase.PerformanceEvaluation --nomapred --presplit=16 --rows=10485760 randomWrite 10

  Single client, 10 threads, writes 100GB data in total
- Test Result
  - Avg latency (in microseconds)
    - Single WAL: 147.16
    - Multiple WAL: 118.18
  - Avg complete time of each thread (in milliseconds)
    - Single WAL: 1552096
    - Multiple WAL: 1248256
- Main settings in hbase-site.xml
  - hbase.hstore.blockingStoreFiles -> 20
  - hbase.hregion.memstore.flush.size -> 402653184
  - hbase.hregion.memstore.block.multiplier -> 4
  - hbase.hstore.compaction.max -> 10
  - hbase.hstore.compaction.min -> 4
  - hbase.hregion.max.filesize -> 8589934592
  - hbase.hregion.majorcompaction -> 604800000
  - hbase.wal.provider -> multiwal (Multiple wal specified)
  - hbase.wal.regiongrouping.numgroups -> 4 (Multiple wal specified)

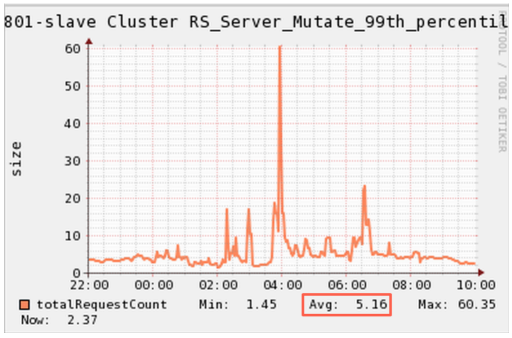
During the testing we made sure enough flush/compaction happens. From the result we could see there's a **~20% improvement** with multiple wal

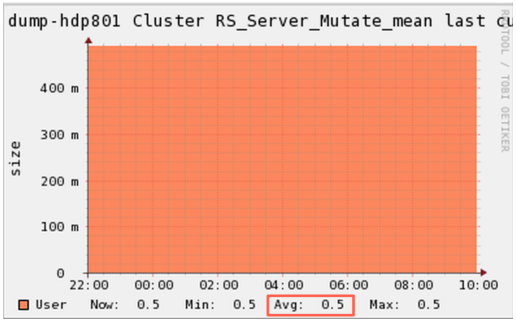
Test Result with pure SATA-SSD disks

Below are the Ganglia monitoring data of our online production cluster, which has 800+ nodes. The cluster serves mixed workloads – heavy reads/writes from index building and machine learning (parameter server) jobs
The hardware/software environment is similar to the testing environment; the only difference is that each node has 3 SAMSUNG SATA-SSD disks and only 9 SATA disks in product cluster (the *pure SATA-SSD* in title means all WAL writes are on SSD, not all disks are SSD, this will be further explained later in this section), and the hbase version is based on 0.98.12 (but the main codes of multiple WAL part are the same).

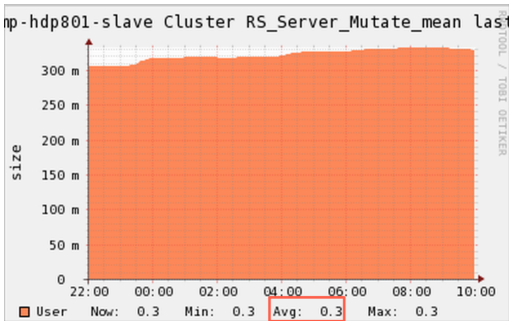


V.S.





V.S.



The left graphs are w/ single wal and the right ones w/ multiple wal, from the results we could see there're a larger (40%) improvement
Notice that we also take advantage of the [HDFS heterogeneous feature] and set WAL directory to ALL_SSD policy, so all writes on WAL are placed on SSD thus further reduce the impact of flush/compaction

Test result with PCIE-SSD disks

We also tested PCIE-SSD with multiple WAL plus sync writes (modifying ProtobufLogWriter#sync to call hsync instead of hflush). Since the random IO performance of PCIE-SSD is about 10 times better than SATA-SSD, and using multiple WAL (more IO pipes) could take better advantage of this feature, we gave it a try on hsync, and get a very promising result.
We also used a more formal and commonly accepted benchmark – YCSB in the

testing. Test settings are as follows:

- One single region server, 128 handlers
- 3 data nodes
- 8 client nodes, 2 YCSB processes on each node, 16 threads in each process
- 16 columns per row, cell size 128B, 2 million records for each thread
- Pre-split 90 regions
- DROP_CACHE means we keep running below command in the background to drop OS cache:
  echo 3 | sudo tee /proc/sys/vm/drop_caches

Below are the results:

RW_MODE	COMPRESS	DROP_CACHE	MULTI HLOG	SYNC_WRITE	THROUGHPUT	AVERAGELATENCY	THPERCENTILELATENCY99
Write	SNAPPY	Yes	4	No	27k	INSERT=4.8ms	INSERT=11.2ms
		No	4	No	30k	INSERT=4.2ms	INSERT=10.0ms
		No	4	Yes	23k	INSERT=5.5ms	INSERT=10.1ms
	NONE	Yes	4	No	27k	INSERT=4.8ms	INSERT=13.4ms
		No	4	No	28k	INSERT=4.6ms	INSERT=11.9ms
Mix(RW7:3)	SNAPPY	Yes	4	No	20k	READ=7.6ms INSERT=4.1ms	READ=113.5ms INSERT=90.5ms
		No	4	No	29k	READ=3.9ms INSERT=5.6ms	READ=69.3ms INSERT=74.6ms
		No	4	Yes	20k	READ=4.3ms INSERT=11.8ms	READ=139.2ms INSERT=182.4ms
	NONE	Yes	4	No	19k	READ=8.8ms INSERT=2.4ms	READ=21.0ms INSERT=7.9ms
		No	4	No	39k	READ=2.5ms INSERT=5.0ms	READ=8.8ms INSERT=20.9ms

The PCIE-SSD we used is similar to FusionIO products (just w/ some improvements in the IO scheduling kernel module); the other hardware and hbase-site.xml settings are the same with other tests mentioned in previous sections. From the result we could see that with multiple WAL and PCIE-SSD, both throughput and latency with hsync are acceptable, even for product usage.

Future Work

Features we could add based on current multiple WAL implementations include (but not limited to):

1. Per-namespace replication
2. Replication monitoring improvements for multiple wal
3. Multiple wal plus HBase-specified outputstream like HBASE-14790 to further improve throughput/latency

Conclusions

Performance data of multiple WAL in both pure SATA disk testing environment and online SSD-SATA mixed disk environment shows obvious improvements on latency/throughput compared to single wal, and multiple wal with PCIE-SSD makes hsync latency promising. All these data demonstrates that multiple wal solution is ready for product usage. Besides, namespace-level write isolation could also be achieved through it. We believe more benefits are waiting to be discovered, and hope this feature could really help and get used in more real clusters. We also need everyone's help on improving it for better use in the future.