Async HDFS Performance Report

Xiaobing Zhou, Tsz Wo Nicholas Sze

Introduction

This report aims to provide an overview of performance of asynchronous calls based on benchmark. Specifically, taking rename with overwrite as an example in the benchmark, performance speedup is recorded.

Experiment Setup

Benchmark client setup:
HADOOP_CLIENT_OPTS=-Xss4096k
HADOOP_HEAPSIZE=5000

Benchmark client hardware:
Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz (32 Virtual Processors, 2 CPUs, 8 cores per CPU) with HT (Hyper-Threading) enabled

	total	used	free	shared	buff/cache	available
Mem:	251G	1.5G	30G	65M	219G	249G
Swap:	4.0G	0B	4.0G			

HDFS setup:
One NameNode and two DataNodes are deployed on three bare metal machines. NameNode is configured with dfs.permissions = false and dfs.namenode.handler.count = 300. DataNodes are configured with dfs.datanode.handler.count = 300. All other configuration otherwise are default. See also the repo to get hadoop configuration and source code of the benchmark.

NameNode hardware:
Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz (32 Virtual Processors, 2 CPUs, 8 cores per CPU) with HT (Hyper-Threading) enabled

	total	used	free	shared	buff/cache	available
Mem:	251G	16G	6.0G	57M	228G	234G
Swap:	4.0G	0B	4.0G			

#1 DataNode hardware:
Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz (32 Virtual Processors, 2 CPUs, 8 cores per CPU) with HT (Hyper-Threading) enabled

	Total	used	free	shared	buff/cache	available
Mem:	251G	3.3G	16G	57M	231G	247G
Swap:	4.0G	0B	4.0G			

#2 DataNode hardware:

Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz (32 Virtual Processors, 2 CPUs, 8 cores per CPU) with HT (Hyper-Threading) enabled

	total	used	free	shared	buff/cache	available
Mem:	251G	3.0G	9.9G	57M	238G	247G
Swap:	4.0G	0B	4.0G			

Experiment overview

The benchmark is done in a HDFS cluster with two DataNodes and one NameNode. All workloads are generated and issued from another separate node (i.e. benchmark client machine). There are five workloads such as 5000, 10000, 20000 and 40000 rename calls. For every workload, single-threaded synchronous, asynchronous as well as multithreaded synchronous calls are invoked. They share the same program path to maximum extent to minimize side effects. Each case is run within a single JVM launch after another is done to make sure they are independent. The total time elapsed in seconds are recorded. For simplicity of terminology, single-threaded sync, async, and threaded sync are used in the Fig. 1, Fig. 2, Fig. 3 and Fig. 4. Specifically, 500 threaded sync means 500 threads of concurrent sync calls.

Regarding the number of threads, 1000 threaded sync call does not mean 1000 threads are explicitly created, instead, 1000 is the size of thread pool (e.g. ExecutorService pool = Executors.newFixedThreadPool(1000);). The actual number of threads or active threads at runtime is affected by computation capacity, scheduling, congestion of shared RPC connection and the extent NameNode is saturated. The actual number of threads might be smaller than the requested one.

Performance Evaluation

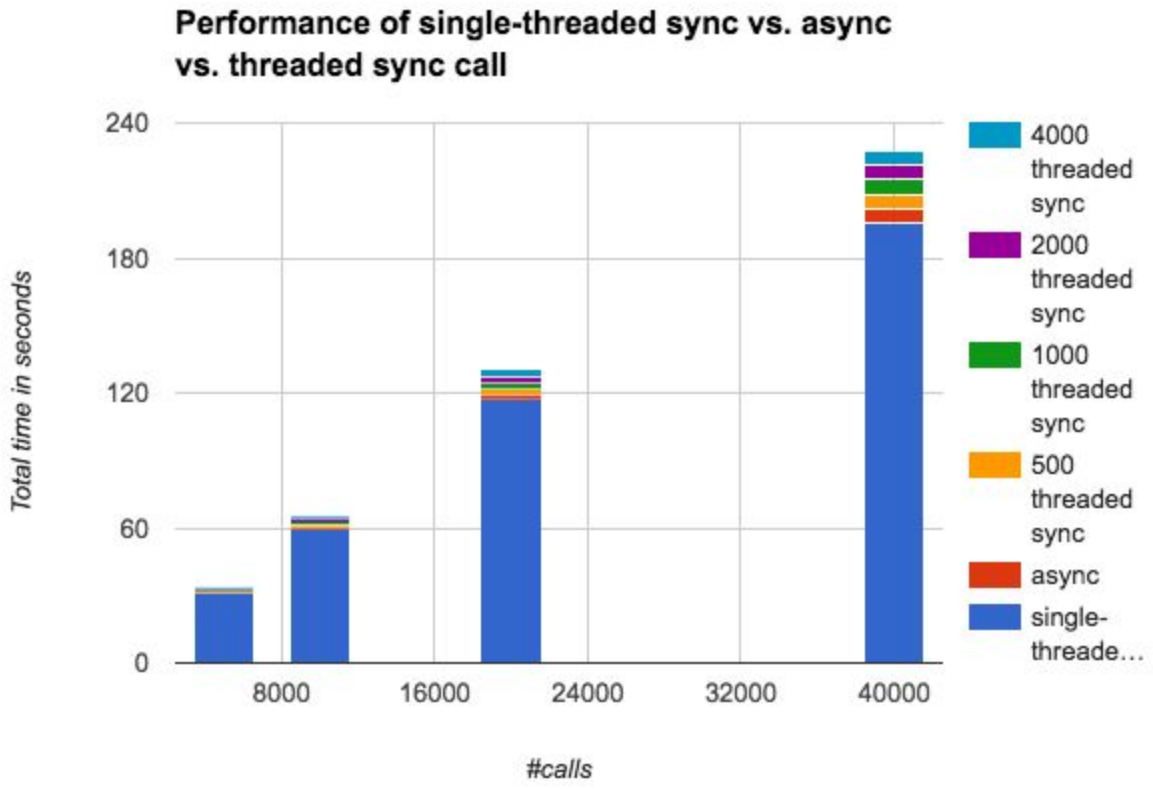

Fig. 1 Performance

Fig. 1 shows total time of running single-threaded sync, async and threaded sync calls at scale of 5000, 10000, 20000 and 40000 HDFS renames. Sync calls dominate the column. Async and threaded sync calls have quite close performance but async is faster.

Tab. 1 and Fig. 2 demonstrate performance speedup resulted from async and threaded sync compared with single-threaded sync call as as a basis.

#calls	async	500 threaded sync	1000 threaded sync	2000 threaded sync	4000 threaded sync
5000	66.85	60.05	52.6	48.36	38.87
10000	58	51.86	51.7	48.94	42.21
20000	49.47	47.8	43.68	42.7	40.42
40000	31.95	30.67	30.31	29.64	29.65

Tab. 1 Performance Speedup compared with single-threaded sync call as a basis

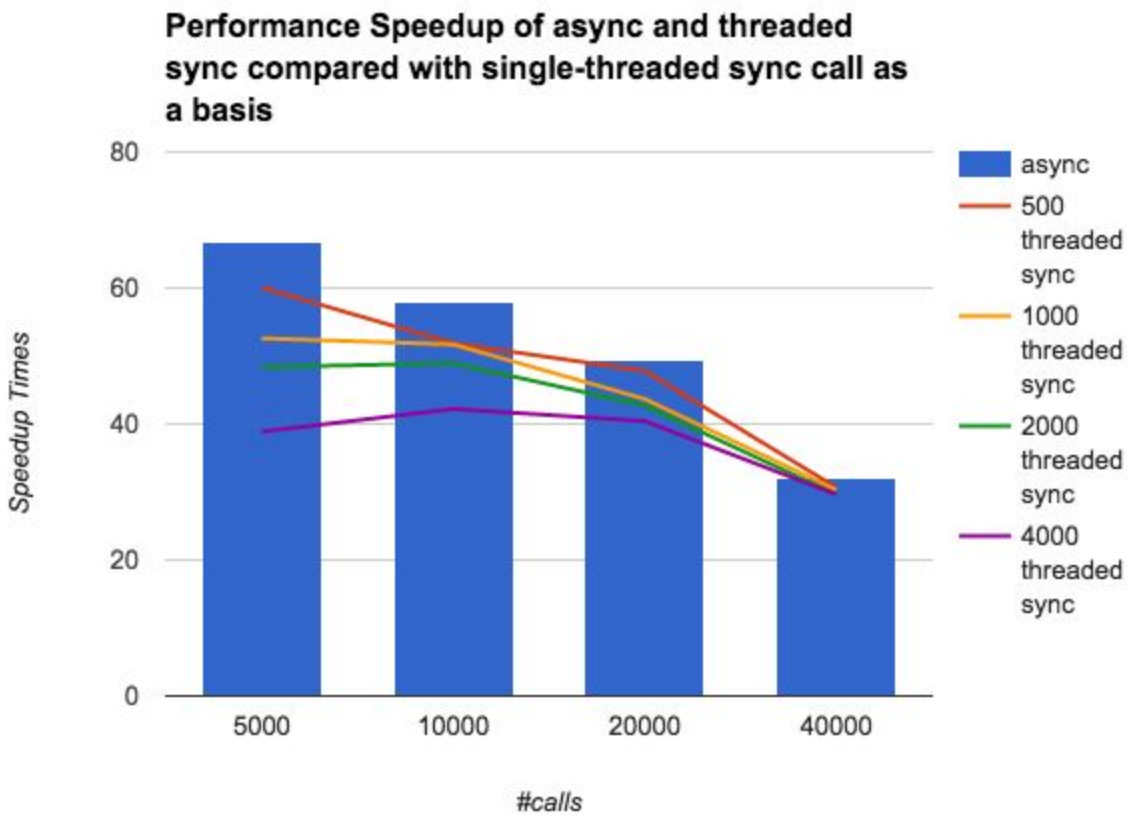

Fig. 2 Performance Speedup

Some facts are observed:

1. Async call is the fastest, it delivered 67, 58, 49 and 32 X speedup at scale of 5000, 10000, 20000 and 40000 calls.
2. In all cases, the larger number of calls tested, the less speedup. This could be because larger number of calls saturated NameNode for longer time. The amortized speedup decreased. For example, the difference of speedup at scale of 40000 calls for all cases are quite close.
3. 500 threaded sync has the best performance (e.g. 60 to 31 X speedup at scale of 5000 to 40000) among all threaded cases, however, it shows up to 10% less speedup than async (e.g. 67 to 32 X speedup at scale of 5000 to 40000)
4. Increased number of threads does not necessarily increase speedup, it however decreases. For example, 60, 53, 48 and 39 X speedup are delivered for 500, 1000, 2000, 4000 threads at scale of 5000 calls. This is possibly because, the larger number of threads, the higher overhead of context switch.