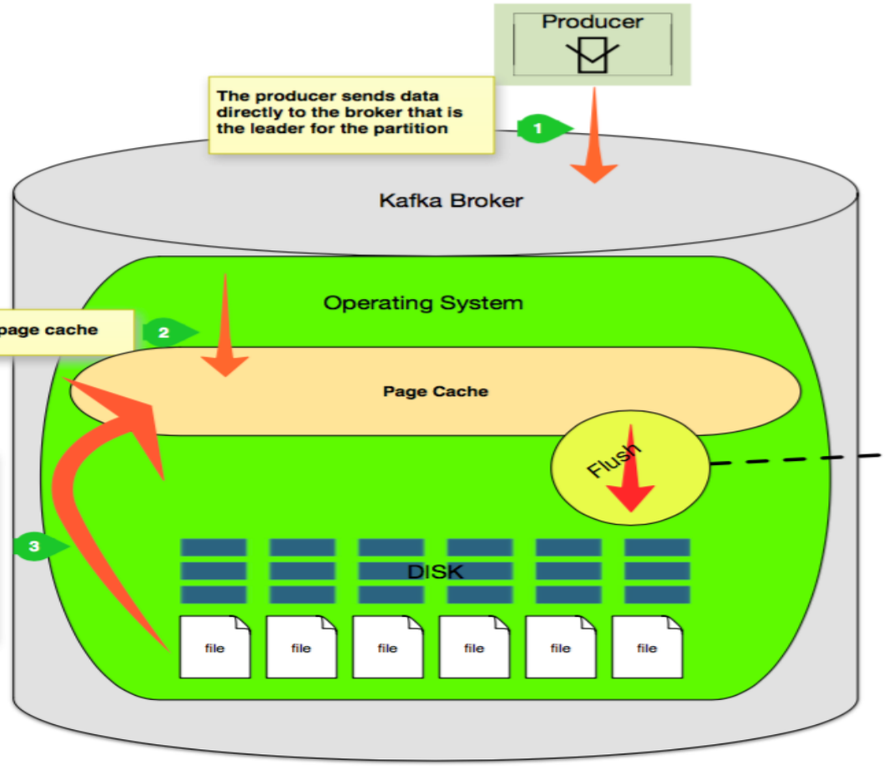
Kafka sizing artifacts

Kafka Write Path



Pages in the main memory that have been modified during writing data to disk are marked as "dirty" and have to be flushed to disk before they can be freed. When a file write occurs, the page backing the particular block is looked up. If it is already found in the page cache, the write is done to that page in the main memory. Otherwise, when the write perfectly fails on page size boundaries, the page is not even read from disk, but allocated and immediately marked dirty. Otherwise, the page(s) are fetched from disk and requested modifications are done.

—Wiki

Not exhaustive List
More info here
http://kafka.apache.org/documentation.html

The number of messages written to a log partition before flushed to disk (log.flush.interval.messages)

The log for a topic partition is stored as a directory of segment files. This setting controls the size to which a segment file will grow before a new segment is rolled over in the log. or per topic - log.segment.bytes.per.topic

OS may flush the page cache. This configuration is OS specific. Ie pdflush

Understanding Linux OS Flush Behavior

In Linux, data written to the filesystem is maintained in pagecache until it must be written out to disk (due to an application-level fsync or the OS's own flush policy). The flushing of data is done by a set of background threads called pdflush (or in post 2.6.32 kernels "flusher threads").
Pdflush has a configurable policy that controls how much dirty data can be maintained in cache and for how long before it must be written back to disk. This policy is described here. When Pdflush cannot keep up with the rate of data being written it will eventually cause the writing process to block incurring latency in the writes to slow down the accumulation of data.

You can see the current state of OS memory usage by doing

cat /proc/meminfo
The meaning of these values are described in the link above.
Using pagecache has several advantages over an in-process cache for storing data that will be written out to disk:

The I/O scheduler will batch together consecutive small writes into bigger physical writes which improves throughput.
The I/O scheduler will attempt to re-sequence writes to minimize movement of the disk head which improves throughput.
It automatically uses all the free memory on the machine

—Apache Kafka dot org

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

Average size of each message

- Account for during disk size. Average Message size + Retention Period * Replication factor
- BB Based Exemple: 1TB + 4 Days * 3 Factor
- Message size will affects network bandwidth
  - For high performance kafka cluster use 10GbE cards

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

Page Cache (Memory) Flushing

- Kafka broker will always write to page cache (OS) first
- Messages are flushed by based on several configurations
  - http://kafka.apache.org/documentation.html
- Flush message to disk controlled by log.flush.interval.message
  - Defaults to Long.MAX_VALUE which is very big
- The number of messages written to a log partition before we force an fsync on the log
  - log.flush.interval.message
- The per-topic override for log.flush.interval.messages
  - log.flush.interval.ms.per.topic
- OS will flush (pdflush) regardless of kafka params
  - https://en.wikipedia.org/wiki/Page_cache
- Default flush configurations will cover most use cases as the durability is provided by replication

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

Disk Type

- We recommend using multiple drives to get good throughput and not sharing the same drives used for Kafka data with application logs or other OS filesystem activity to ensure good latency.
- Since Kafka has replication the redundancy provided by RAID can also be provided at the application
- Isolate disk and only use to store the application data
- Using multi disk with multi-directory often resulted in better performance than using RAID
- RAID can potentially do a better load balancing of the data at the low-level. But the major downside of RAID is usually a big performance hit for write throughput and reduces the available disk space.
- Another potential benefit of RAID is the ability to tolerate disk failures. Rebuilding the RAID array is so I/O intensive that it effectively disables the server, so this does not provide much real availability improvement.

- https://kafka.apache.org/08/ops.html

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Number of Partitions for a Topic

- Number of topics and partitions impact how much can be stored in page cache
- Topic/Partition is unit of parallelism in Kafka
- Partitions in Kafka drives the parallelism of consumers
- Throughput requirements drive number of number of partitions on a topic.
- Formula
  - P= Throughput from producer to a single partition is
    - https://github.com/apache/kafka/blob/trunk/bin/kafka-producer-perf-test.sh
  - C= Throughput from a single partition to a consumer
    - https://github.com/apache/kafka/blob/trunk/bin/kafka-consumer-perf-test.sh
  - T= Target throughput
  - Required # of partitions = Max (T/P, T/C)

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

Number of Partitions for a Topic - Example

- Producer is able to insert 100 messages per second into the single partition
  - ie Tested via https://github.com/apache/kafka/blob/trunk/bin/kafka-producer-perf-test.sh
- Consumer is able to read from 200 messages per second
  - ie Test via https://github.com/apache/kafka/blob/trunk/bin/kafka-consumer-perf-test.sh
- A target throughput as been identified as 1000 messages per second
- Number of partitions required 10 = Max(1000/100, 1000/200)

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Latency

- More partitions can increase the latency
- The end-to-end latency in Kafka is defined by the time from when a message is published by the producer to when the message is read by the consumer.
- Kafka only exposes a message to a consumer after it has been committed
  - i.e. when the message is replicated to all the in-sync replicas
- Replication 1000 partitions from one broker to another can take up 20ms.
  - This can be too high for some real-time applications
- In new Kafka producer messages will be accumulated on the producer side
  - It allows users to set upper bound on the amount of memory used for buffering incoming messages
  - Internally producer buffers the message per partition
  - After enough data has been accumulated or enough time has passed, the accumulated messages will be removed and sent to the broker
- More partitions = More messages that will be accumulated producer side
- Similarly on the consumer side it fetches batch of messages per partitions
  - The more partitions that consumer is subscribing to the more memory it needs

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

Replicas & Type of Acknowledgment

- Replicas on Kafka do not perform any functionality outside of provide HA availabilities.
- Replicas Do not service reads or writes
- Play a role in during producers acknowledgements
  - Async – Producer sends message to broker (Leader) and a acknowledgment is sent back to producer once broker receives message
  - Sync - Producer sends message to broker (Leader) and a acknowledgment is sent back to producer once message has been received by all replicas. Performance impact
- Include replica factor in amount of disk storage required on brokers

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

Message Guarantee

- So effectively Kafka guarantees at-least-once delivery by default and allows the user to implement at most once delivery by disabling retries on the producer and committing its offset prior to processing a batch of messages. Exactly-once delivery requires co-operation with the destination storage system but Kafka provides the offset which makes implementing this straight-forward.

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

Sizing Artifacts Required

- For each topic
  - Average size of message
  - Throughput Requirements (will drive # of partitions)
  - Replication Factor
  - Retention period
- Linkedin publish benchmarks
  - https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines
  - Single producer thread, no replication
    - 821,557 records/sec(78.3 MB/sec)
  - Single producer thread, 3x asynchronous replication
    - 786,980 records/sec(75.1 MB/sec)

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Kafka Hardware Profile

- 16/32 core dual socket (16CPU)
- 128/256 Gig of Ram
  - 32GB Kafka JVM
  - Rest for OS + Page Cache
- 8x2TB disk
  - EXT4
    - XFS may handle locking during fsync better. EXT4 mostly used in field
  - SAS or SSD preferred
  - Based on Retention period
  - Account for Broker and # of partitions it will handle
  - JBOD – Assuming you will use replication
    - Optional RAID10 - The primary downside of RAID is that it is usually a big performance hit for write throughput and reduces the available disk space.
- 10GigE
  - Bonded NICs for extreme performance

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Appendix

- Producer performance testing
  - bin/kafka-run-class.sh org.apache.kafka.tools.ProducerPerformance --topic first --num-records 50000000 --record-size 100 --throughput -1 --producer-props acks=1 bootstrap.servers=localhost:9092 buffer.memory=67108864 batch.size=8196

© Hortonworks Inc. 2011 – 2016. All Rights Reserved