Factors impacting Kafka performance CPU

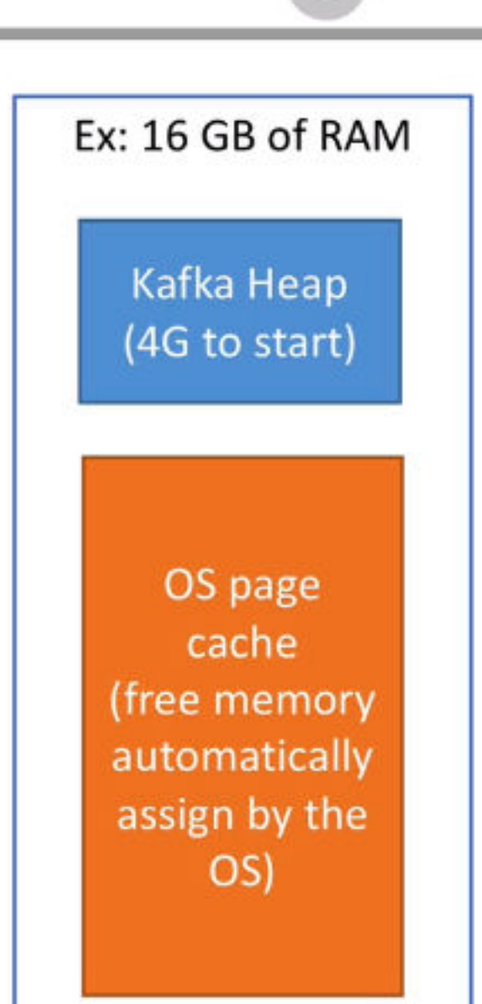
- CPU is usually not a performance bottle neck in Kafka because Kafka does not parse any messages , but can become one in some situations
- If you have SSL enabled, Kafka has to encrypt and decrypt every payload, which adds load on the CPU
- Compression can be CPU bound if you force Kafka to do it. Instead, if you send compressed data, make sure your producer and consumers are the ones doing the compression work (that's the default setting anyway)
- Make sure you monitor Garbage Collection over time to ensure the pauses are not too long

Factors impacting Kafka performance Disks: I/O

- Reads are done sequentially (as in not randomly), therefore make sure you should a disk type that corresponds well to the requirements
- Format your drives as XFS (easiest, no tuning required)
- If read / write throughput is your bottleneck
  - it is possible to mount multiple disks in parallel for Kafka
  - The config is log.dirs=/disk1/kafka-logs,/disk2/kafka-logs,/disk3/kafka-logs…
- Kafka performance is constant with regards to the amount of data stored in Kafka.
  - Make sure you expire data fast enough (default is one week)
  - Make sure you monitor disk performance

Factors impacting Kafka performance RAM

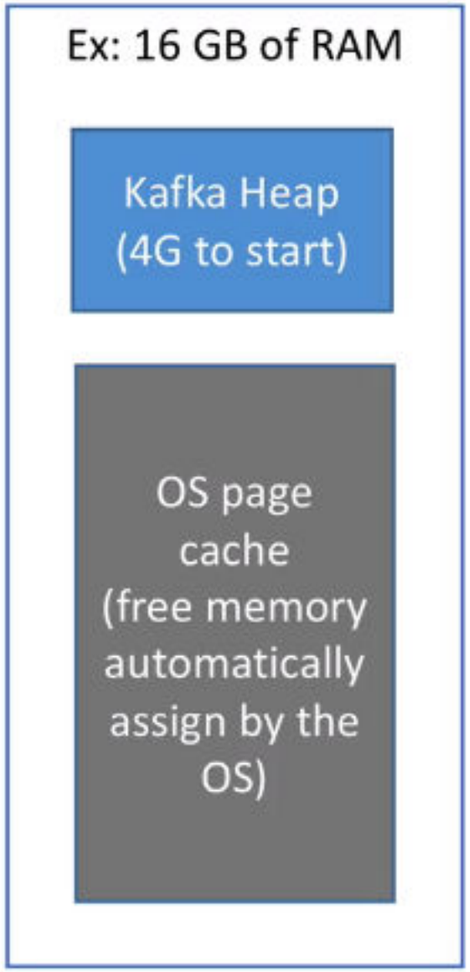
- Kafka has amazing performance thanks to the page cache which utilizes your RAM
- Understanding RAM in Kafka means understanding two parts:
  - The Java HEAP for the Kafka process
  - The rest of the RAM used by the OS page cache



- Let's understand how both of those should be sized
- Overall, your Kafka production machines should have at least 8GB of RAM to them (the more the better – it's common to have 16GB or 32GB per broker)

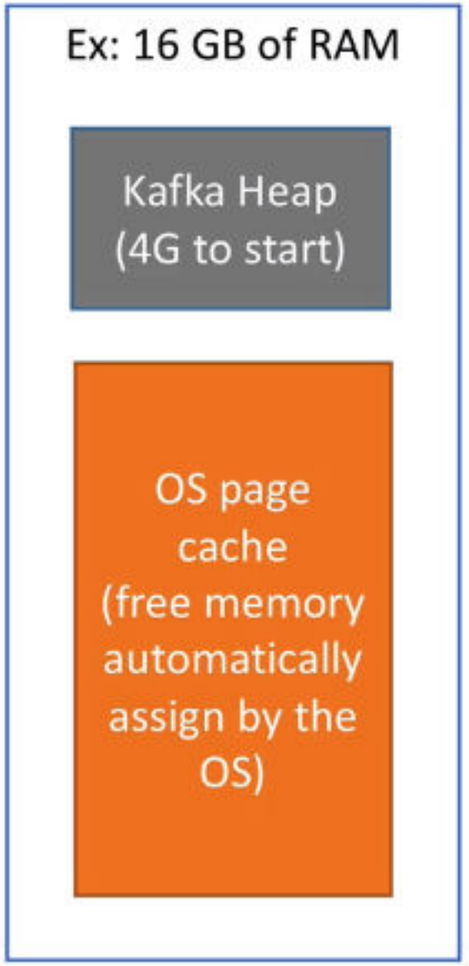
Factors impacting Kafka performance
RAM – Java Heap

- When you launch Kafka, you specify Kafka Heap Options (KAFKA_HEAP_OPTS environment variable)
- I recommend to assign a MAX amount (-Xmx) of 4GB to get started to the kafka heap:
- `export KAFKA_HEAP_OPTS="-Xmx4g"`
- Don't set –Xms (starting heap size):
  - Let heap grow over time
  - Monitor the heap over time to see if you need to increase Xmx
- Kafka should keep a low heap usage over time, and heap should increase only if you have more partitions in your broker



Factors impacting Kafka performance
RAM – OS Page Cache

- The remaining RAM will be used automatically for the Linux **OS Page Cache**.
- This is used to buffer data to the disk and this is what gives Kafka an amazing performance
- You don't have to specify anything!
- Any un-used memory will automatically be leveraged by the Linux Operating System and assign memory to the page cache
- Note: Make sure swapping is disabled for Kafka entirely **vm.swappiness=0 or vm.swappiness=1 (default is 60 on Linux)**



Factors impacting Kafka performance Network

- Latency is key in Kafka
  - Ensure your Kafka instances are your Zookeeper instances are geographically close!!!
  - Do not put one broker in Europe and the other broker in the US
  - Having two brokers live on the same rack is good for performance, but a big risk if the rack goes down.
- Network bandwidth is key in Kafka
  - Network will be your bottleneck.
  - Make sure you have enough bandwidth to handle many connections, and TCP requests.
  - Make sure your network is high performance
- Monitor network usage to understand when it becomes a bottleneck

Factors impacting Kafka performance
Operating System (OS)

- Use Linux or Solaris, running production Kafka clusters on Windows is not recommended.
- Increase the file descriptor limits (at least 100,000 as a starting point) https://unix.stackexchange.com/a/8949/170887
- Make sure only Kafka is running on your Operating System. Anything else will just slow the machine down.

Advanced Kafka Configuration Important parameters to be aware of

- auto.create.topics.enable=true => set to false in production
- background.threads=10 => increase if you have a good CPU
- delete.topic.enable=false => your choice
- log.flush.interval.messages => don't ever change. Let your OS do it
- log.retention.hours=168 => Set in regards to your requirements
- message.max.bytes=1000012 => increase if you need more than 1MB
- min.insync.replicas=1 => set to 2 if you want to be extra safe
- num.io.threads=8 => ++if your network io is a bottleneck
- num.network.threads=3 => ++if your network is a bottleneck

Advanced Kafka Configuration Important parameters to be aware of

- num.recovery.threads.per.data.dir=1 => set to number of disks
- num.replica.fetchers=1 => increase if your replicas are lagging
- offsets.retention.minutes=1440 => after 24 hours you lose offsets
- unclean.leader.election.enable=true => false if you don't want data loss
- zookeeper.session.timeout.ms=6000 => increase if you timeout often
- broker.rack=null => set your to availability zone in AWS
- default.replication.factor=1 => set to 2 or 3 in a kafka cluster
- num.partitions=1 => set from 3 to 6 in your cluster
- quota.producer.default=10485760 => set quota to 10MBs
- quota.consumer.default=10485760 => set quota to 10MBs

Factors impacting Kafka performance
Other

- Make sure you have enough file handles opened on your servers, as Kafka opens 3 file descriptor for each topic-partition-segment that lives on the Broker.
- Make sure you use Java 8
- You may want to tune the GC implementation: (see in the resources)
- Set Kafka quotas in order to prevent unexpected spikes in usage

Running Kafka on AWS in Production

- Separate your instances between different availability zones
- Use st1 EBS volumes for the best price / performance ratio
- Mount multiple EBS volumes to the same broker if you need to scale
- Use *r4.xlarge* or *m4.2xlarge* if you're using EBS (these instances are EBS optimized). You can use something smaller but performance may degrade
- Setup DNS names for your brokers / fixed IPs so that your clients aren't affected if you recycle your EC2 instances