CHAPTER 12

Cost Reduction of Kafka Clusters

This chapter zeroes in on the pivotal aspect of reducing hardware costs in Apache Kafka clusters. By carefully examining the choices between deploying Kafka on the cloud and on-premises, we'll delve into strategies that can lead to significant cost reduction.

The chapter begins by exploring the vital aspects that influence the scaling of Kafka clusters, with particular attention to Kafka on-premises. Here, we'll outline various options and their direct impact on hardware costs, drawing comparisons between cloud and on-premises solutions. The technical, managerial, and financial facets are analyzed, all within the context of achieving cost savings.

As we progress, the chapter takes a deep dive into the hardware considerations that play a central role in controlling costs. This includes detailed examinations of RAM, CPU cores, disk storage, and IOPS, with an emphasis on identifying the most economical configurations and setups. Practical examples are provided to clarify the optimal hardware selection for both cloud and on-premises deployments, with clear insights into the tradeoffs involved.

The concluding section presents a series of real-world examples that vividly illustrate proven strategies for cost reduction in over-provisioned Kafka clusters. These include specific findings, options for reducing costs, and recommendations tailored to different scenarios and cluster specifications. Special attention is given to cloud deployments, but the principles can be applied more broadly.

Through these examples and the detailed exploration throughout the chapter, you'll gain a concrete understanding of how to minimize the hardware costs of your Kafka cluster. The insights offered are not merely theoretical; they are drawn from real-world applications and are designed to empower you to make informed, cost-effective decisions.

E. Eldor, *Kafka Troubleshooting in Production*, https://doi.org/10.1007/978-1-4842-9490-1_12

Determining If You Can Reduce Costs in Your Kafka Clusters

When it comes to scaling a Kafka cluster, the deployment environment makes a crucial difference. Specifically, scaling is often much more straightforward when the cluster is on the cloud as compared to on-premises. This difference stems from both technical and managerial reasons that make scaling an on-premises Kafka cluster more complex.

In discussing the scaling of an on-premises cluster, we need to take into consideration three key aspects: technical, managerial, and financial. Scaling the cluster in this context refers to making adjustments that may include adding or removing brokers or changing the type of the brokers. These changes can encompass variations in RAM, CPU cores, disk storage, or even disk type.

To illuminate the distinction between deploying on the cloud and on-premises, the following sections delve into the various reasons for scaling a cluster and explore how this process can be accomplished in both environments.

Lack of RAM

Managing memory in a Kafka cluster is essential. In the cloud, you can easily add more RAM by choosing bigger instances. But for clusters deployed on-premises, it's a bit more complex. You can either add or swap out memory sticks, or just bring in more machines. Each choice has its own set of pros and cons, so it's crucial to think it through.

Cloud-Based Cluster

In order to add more RAM to a Kafka cluster that is deployed on the cloud, we just need to spin off a new cluster with new instances that have more RAM.

On-Prem Cluster

When aiming to add more RAM to a Kafka cluster deployed on-premises, we have several possible approaches. If there are available memory slots on the motherboard, additional DIMMs can be added to each broker. Alternatively, if the size of the current DIMMs deployed on the motherboard is smaller than the maximal size, these can be replaced with larger ones.

If all memory slots are occupied with DIMMs at their maximal size, adding more machines becomes the viable option. It's important to note that the term DIMM used here refers to a memory module or memory stick. This context is specifically referring to DDR4 DIMMs, which range in size from 8GB to 64GB.

Discussion

There are several courses of action when dealing with a Kafka cluster that's deployed on-prem and lacks RAM, and choosing which way to go depends on the use case. If there are available slots for DIMMs, then the easiest way is to add DIMMs. If that solves the issue, then we're done.

However, what if it doesn't solve our problem? In such case, we have two options, as discussed next.

Replace the DIMMs with Larger DIMMs That Have More RAM

This strategy can be employed if the current size of the DIMMs isn't at its maximum. Replacing all the DIMMs in all brokers with larger-sized DIMMs has specific advantages and challenges.

From a technical standpoint, this approach is correct, as it merely involves adding RAM without introducing additional brokers. This means there's no need to reassign the topics since the number of brokers remains the same.

However, the financial implications of this solution must be considered. For instance, purchasing a single server with 24 DIMMs of size 32GB may cost between two to five times more than purchasing just the DIMMs themselves, depending on the server type.

The managerial aspect also presents challenges, making it a tough decision to undertake.

This process involves removing all existing DIMMs from the brokers, replacing them with the larger ones, and storing the old DIMMs elsewhere. Data center owners often resist purchasing hardware that will not be utilized (such as the old DIMMs), making this solution less attractive from an operational perspective.

If the current size of the DIMMs isn't the maximal one, we can replace all the DIMMs in all brokers with DIMMs of a bigger size.

Scale Out by Adding More Brokers with the Same Amount of RAM

This is an alternative solution to increasing the cluster's capacity. From a managerial standpoint, this method has the advantage of not requiring the disposal of old DIMMs, making it an easier decision to implement. Moreover, adding more machines to the cluster is often a simpler task compared to replacing all the DIMMs in the current brokers.

However, this approach does have its drawbacks. If the DIMMs in the current brokers can be replaced with larger ones, adding more brokers with the same amount of RAM might end up being more costly. Essentially, the financial implications could outweigh the convenience, especially if the current DIMMs have not reached their maximum potential size. Therefore, careful consideration of the technical requirements and the budget constraints should guide the decision-making process.

The decision whether to replace the DIMMs or add more brokers needs to take into account all these aspects—technical, financial, maintenance and managerial. In order to make the right call, you'll need to put into your calculation the importance of each aspect, and according to that decide on the correct path.

Lack of CPU Cores

Cloud-Based Cluster

When addressing the lack of CPU cores in a Kafka cluster deployed on the cloud, there are two main strategies that we can pursue: scale up or scale out.

- For the scale-up approach, we can create a new cluster with the same number of instances but select instance types that come with more cores. This essentially enhances the existing configuration with additional processing power without increasing the number of instances.
- The scale-out method involves adding more instances (with the same instance type of the the existing instances) to the cluster. This expands the cluster's size without changing the individual processing capabilities of each instance, providing a broader rather than deeper enhancement.

On-Prem Cluster

When an on-premises Kafka cluster requires more CPU cores, two primary strategies can be considered: scaling up or scaling out.

- The scale-up approach involves creating a new cluster by replacing the existing machines with new ones that have more CPU cores. This strategy has the advantage of ensuring that the cluster will have the necessary cores, but it comes with financial and managerial challenges. For example, the old machines, now unused, represent an additional cost, and explaining to the data center owner why a cluster was initially provisioned with insufficient CPU cores can be challenging.
- The scale-out approach, on the other hand, entails adding more brokers of the same type to the existing cluster. This method avoids the need to remove current brokers, making it more palatable from both a financial and managerial perspective. However, it may lead to other complexities, such as ending up with a cluster that meets CPU requirements but has excessive RAM, disk storage, or disk IOPS.

Discussion

When facing the decision whether to scale out or scale up a Kafka cluster to get more CPU cores, various considerations come into play, and the optimal approach might differ between on-premises and cloud-based deployments.

In the context of an on-premises cluster, scaling out often seems more financially attractive, as it can extend existing resources without necessitating the purchase of entirely new or costlier hardware. Conversely, in a cloud-based environment, financial considerations may be less significant, as replacing machines doesn't result in the retention of old, unused hardware.

From a managerial perspective, scaling out an on-prem cluster may also be more appealing. It avoids the disposal or replacement of current equipment, aligning more closely with the existing infrastructure and investment, and making the decision-making process smoother. This managerial concern is typically not relevant for a cloud-based cluster, where hardware replacement is a transparent operation.

Technically, scaling out is a more straightforward solution for both on-prem and cloud-based clusters. It allows for the necessary expansion without the need to migrate topics from the old cluster. Reassignment of partitions to new brokers simplifies the process, making scaling out a commonly preferred method across both deployment scenarios.

Lack of Disk Storage

Cloud-Based Cluster

When adding more disk storage to a Kafka cluster deployed on AWS, it's crucial to consider the underlying storage technology, namely Elastic Block Store (EBS) and NVMe (Non-Volatile Memory Express) devices. Each of these options has unique characteristics and considerations.

With EBS, you have the flexibility to attach volumes to existing instances, allowing you to increase disk space without altering the existing infrastructure. You can also scale out the cluster by adding more brokers with the same amount of EBS disk space, distributing the data and load across a more extensive set of nodes. If needed, you can scale up the cluster by replacing current brokers with instances that have larger EBS volumes. Since EBS offers various types and sizes, you can choose the best fit for performance and cost, bearing in mind that EBS is network-attached storage, which might have implications on latency and IOPS requirements for your Kafka workload.

On the other hand, NVMe devices, known for low-latency and high-throughput storage, offer different opportunities for scaling. If your instances support NVMe, you can attach additional NVMe disks to each broker. This option may be especially beneficial for write-intensive workloads, giving you the option to scale out by adding more brokers equipped with NVMe disks. Alternatively, you can scale up by selecting instances with larger NVMe storage, allowing for a seamless increase in disk space and potential performance benefits. Keep in mind that NVMe is typically local to the instance, so data durability and replication strategies must be meticulously planned.

On-Prem Cluster

In an on-premises environment, adding more disk storage to a Kafka cluster requires careful consideration of the existing hardware configuration, along with identifying the most suitable method for expansion. The term *storage drawer,* which refers to the component within a server chassis that houses disk drives, is pivotal in this context.

One strategy to increase storage involves replacing the current disk drives in the storage drawers with those having more storage capacity. This approach enhances the existing infrastructure without necessitating additional hardware, capitalizing on the opportunity to upgrade without substantial changes.

If there are unused slots in the storage drawers, then adding more disk drives can be a viable option. This approach makes the most of existing capacity in the infrastructure, promoting a cost-effective increase in storage without transforming the overall hardware configuration.

Alternatively, the cluster can be expanded by adding more brokers, each equipped with the same number and type of disks.

Discussion

When confronting the challenge of adding more disk storage to a Kafka cluster, various factors must be examined, and the most suitable approach may differ between cloud-based and on-premises deployments.

In the cloud-based environment, particularly with AWS, the availability of different underlying storage technologies, such as Elastic Block Store (EBS) and NVMe, provides flexibility in scaling. With EBS, you can extend disk space effortlessly, offering options to scale out by adding more brokers or scale up by choosing larger storage volumes. However, considerations regarding network latency and IOPS requirements must be factored in. On the other hand, NVMe devices offer low-latency and high-throughput storage, but data durability and replication strategies must be carefully planned, as NVMe is typically local to the instance.

For an on-premises cluster, the decision-making process requires a thorough understanding of existing hardware configurations. Options might include replacing existing disk drives with larger ones, adding more drives if slots are available, or expanding the cluster with more brokers with the same type of disks. The choices often hinge on financial efficiency and alignment with the existing infrastructure, along with performance requirements. The approach must be consistent with the Kafka workload, ensuring that considerations such as latency, throughput, and data replication are adequately addressed.

From a managerial perspective, decisions regarding scaling disk storage in an on-prem cluster often lean toward maximizing existing resources without unnecessary expenditures on new hardware. This aligns with typical data center ownership concerns, favoring solutions that work with the existing investments. By contrast, the flexibility and variety of options in a cloud-based environment might allow for a more straightforward scaling, but with careful attention to specific storage characteristics and their impact on performance and reliability.

Lack of Disk IOPS

If the disks have reached their maximum capacity for disk IOPS, there are two options to add more IOPS. The first option is adding more disks of the same type or replace the existing disks with ones that have a higher IOPS. You can find more info on that in the "Lack of Disk Storage" section.

However, if the lack of disk IOPS is due to read operations on the disks, the problem may not be lack of disk IOPS, but rather lack of RAM or lagging consumers. This is because the reads are initially performed from the page cache, and only if the messages are not found there, the operating system will read the data from the disks. In this case, instead of adding more disk IOPS to the cluster, it may be beneficial to check whether consumers lag, and/or add more RAM. You can find more information on this in the "Lack of RAM" section.

Cost Optimization Strategies for Kafka Clusters in the Cloud

To reduce the hardware costs associated with a Kafka cluster, two main strategies can be considered: scaling in the cluster by using fewer brokers, or scaling down the cluster by using smaller brokers. Each approach has unique benefits and challenges. The following sections explore examples of over-provisioned Kafka clusters, starting with an in-depth description of the cluster's specifications and hardware resource utilization.

Following that, we'll assess the options for scaling in and scaling down the cluster where relevant. We'll also illustrate the potential cost savings for each strategy, measured as a percentage of current costs, and offer guidance on the best scaling method to minimize costs without compromising the cluster's stability.

Additional Considerations

Before we delve into the specific examples of over-provisioned Kafka clusters, this section goes over several considerations that may influence your decision-making process. From the deployment environment (cloud vs. on-prem) to the technical details such as hyper-threading, CPU types, and normalized load averages, these aspects provide the context for the subsequent analysis.

Cloud vs. On-Prem

The following examples refer only to Kafka clusters that are deployed in the cloud, and the recommendation of how to reduce the costs of these clusters refers only to how to perform this in the cloud and not on-prem.

Hyper-Threading

The number of CPU cores in each example is calculated under the assumption that hyper-threading is enabled. When hyper-threading is enabled, the Linux kernel can create two logical processors (threads) for each physical core, allowing two threads to execute simultaneously on a single core.

For example, consider a server that has two sockets. Each socket has 12 cores, and hyper-threading is enabled in the Linux kernel. This server has 24 physical cores but the OS sees 48 cores due to the hyper-threading. In this case, we'll refer to this server as having 48 cores and not 24, since that's the number of cores that the OS sees.

CPU Type

There are different CPU types available in AWS, such as i386, AMD, and ARM. Each of these types has unique characteristics that may influence the performance and cost of the Kafka cluster. However, the influence that each CPU type has on the Kafka cluster isn't discussed in this chapter.

Normalized Load Average (NLA)

One of the metrics that we'll use in order to indicate the load on the clusters is the Normalized Load Average (NLA), so let's look at how the NLA is computed.

While the *load average* is a measure of the number of processes that are currently running or waiting to run in the CPU queue (for time periods of the last 1, 5, and 15 minutes), the *normalized load average* is a scaled value that represents the load average relative to the number of CPU cores on the system. The value of the NLA is defined as: (Load Average/Normalization Factor) x 100.

For example, consider a machine with eight cores. If the load average for the past five minutes is four, that means that on average there were four processes either in a runnable or waiting state. In that case, the NLA is (4/8) X 100 = 0.5.

Scale In

For the sake of clarity, in all the examples that follow, when we explore options to scale in a Kafka cluster, the sections specifically look at using instance types that belong to the same family as the original brokers. While there is indeed the option to scale in a cluster by using instance types from a different family, this chapter doesn't consider that approach, in order to maintain simplicity in this examination.

Example 1

Table 12-1 shows a Kafka cluster with four brokers.

Table 12-1. *A Kafka Cluster with Four Brokers*

# CPU Cores Per Broker	RAM per Broker	Disk Storage Per Broker (NVMe)	%us	%sy	%wa
48	384GB	30TB (4 disks)	5%	20%	0%

Total CPU% Utilization	Normalized Load Average	Used Disk Space in /var/lib/kafka	Network Processor Idle
25%	0.25	15%	99%

Findings

CPU utilization: The cluster is over-provisioned in terms of CPU (only 25% CPU utilization).

Disk storage usage: The cluster is over-provisioned in terms of disk storage (only 15% disk storage usage).

Disk IOPS utilization: There are no reads from the disks (because the `%wa` is low).

Load on the Cluster: The NLA is far from reaching 1.0, which means that the load on the cluster is low.

Options for Reducing Costs

There are two ways to reduce the hardware costs of the cluster—scale down or scale in.

Let's look at the implications of each scaling option and then compare between the two. This chapter recommends the option that is best in terms of cost reduction and maintenance.

Scale Down

Assuming that you want to remain with the same instance family, you can replace the current brokers with brokers that have half the cores, disk storage, and RAM. The reason is that in AWS, the next instance type that's smaller than the current instances has 24 cores, 15TB storage, and 192GB RAM. Let's check each hardware aspect and see how much you can scale it down:

- *CPU:* Use 24 cores instead of 48. Since even with half the number of cores, the brokers will have CPU utilization of only 50%, 24 cores seems to be enough, given that you have a rough estimation that CPU utilization is linear to the number of CPU cores. This is a valid estimation since the CPU utilization is mostly contributed to `us%` and `sy%` and not to disk wait time (`wa%`).
- *Storage:* In AWS, instances with 24 cores arrive with half of the disk storage (15TB) compared to instances with 48 cores. With this amount of storage, the storage usage will be only 30 percent, which is still low.
- *RAM:* In AWS, instances with 24 cores arrive with half of the RAM (192GB RAM) compared to instances with 48 cores. You can tell whether the RAM is sufficient only by reducing the amount of RAM and then checking if there are more reads from the disks. Since there are currently no reads from the disks, you can use only 192GB RAM and see if it's enough by checking the rate of read IOPS from the disks.

Scale In

Since the cluster has four brokers, the option of scaling in the cluster depends on the replication factor (RF) of the topics in the cluster.

If the RF is 3, it's not recommended to remove even a single broker. The reason is that if a single broker fails in a cluster, only two brokers will be left in the cluster and the requirement of three replicas per topic won't be satisfied.

However if the RF is 2, you can remove a single broker and leave the cluster with three brokers. Even if one broker fails, the cluster will still have two brokers, which means that each topic will still have two replicas and the replication requirement will be met.

Recommendation

Scaling in the cluster would reduce the hardware costs by 50 percent since all the brokers will be replaced with brokers at half the price (and half the hardware resources as well). This will require migrating all the topics to the new brokers without having to change the number of partitions.

On the other hand, scaling down the cluster would reduce the costs by 25 percent, but that's recommended only in case the replication factor of the topics is 2 and not 3. This will require a reassignment of the partitions and a change to the number of partitions of all the topics in which their number of partitions doesn't divide equally by 3.

Example 2

Table 12-2 shows a Kafka cluster with six brokers.

Table 12-2. *A Kafka Cluster with Six Brokers*

# CPU Cores Per Broker	RAM Per Broker	Disk Storage Per Broker (NVMe)	%us	%sy	%wa
8	64GB	5TB (2 disks)	30%	12%	0% with peaks of 4%

Total CPU% Utilization	Normalized Load Average	Used Disk Space in /var/lib/kafka	Network Processor Idle
45%	0.6	4%	60%

Findings

CPU utilization: The cluster is over-provisioned in terms of CPU (only 45% CPU utilization).

Disk storage usage: The cluster is over-provisioned in terms of disk storage (only 4% disk storage usage).

Disk IOPS utilization: Most of the time, the disks aren't utilized, but during the day there are several spikes of reads from the disks. This causes the CPU `wa%` to reach a value of 4%, which is quite high.

Load on the cluster: The NLA is 0.6, which isn't low but also not that high.

Network processor idle percentage: This value is really low, only 60%.

Options for Reducing Costs

A low value for Network Processor Idle indicates some bottleneck in the cluster or that there are just not enough network threads. In a healthy cluster, this value should be 99 percent, and it's surprisingly low given the fact that both CPU usage and load average aren't that high. The spikes in the `wa%` also indicate there's some issue in the cluster that causes the disks to be highly utilized in terms of disk IOPS.

Recommendation

At first sight, this cluster seems over-provisioned in terms of CPU cores, since its CPU utilization is at ±45%. But the low network processor idle option shows that it won't be a smart move to scale down the cluster because it suffers from some issue that might already cause latency for its clients, and reducing cores might make these symptoms worse.

So in this case you should focus on investigating the cause of the problematic symptom instead of reducing the cost of the cluster.

Example 3

Table 12-3 shows a Kafka cluster with 12 brokers.

Table 12-3. *A Kafka Cluster with 12 Brokers*

# CPU Cores Per Broker	RAM Per Broker	Disk Storage Per Broker (NVMe)	%us	%sy	%wa
12	96GB	7.5TB	27%	12%	2%

Total CPU% Utilization	Normalized Load Average	Used Disk Space in /var/lib/kafka	Network Processor Idle
40%	0.5	50%	99%

Findings

CPU utilization: The cluster is over-provisioned in terms of CPU (only 40% CPU utilization).

Disk storage usage: The cluster is over-provisioned in terms of disk storage (only 50% disk storage usage).

Disk IOPS utilization: The `wa%` is 2%, which means there are reads/writes to the disks.

Load on the cluster: The NLA is only 0.5, which isn't high.

Options for Reducing Costs

Scaling Down

The option of scaling down the cluster isn't feasible due to potential CPU saturation. The reason is that the only way in AWS to scale down is to use instances with 24 cores. With half of the current cores, the brokers will have CPU utilization of 80-85 percent, which might sometimes lead to a load average of more than 1.0 and to latency in the clients.

Scaling In

Scaling in the cluster involves reducing it from 12 brokers to 8. This reduction is expected to affect your CPU and storage utilization as follows: CPU usage may rise from 40 to 60 percent, although it's worth noting that CPU utilization doesn't always scale linearly. The storage usage is expected to go from 50 to 75 percent. By implementing this change, you can anticipate a 30 percent savings on hardware costs without encountering any CPU

or storage limitations. To accomplish this transition, you need to reassign the partitions across all the topics and adjust the number of partitions to ensure an even distribution among the remaining brokers.

Example 4

Table 12-4 shows a Kafka cluster with ten brokers.

Table 12-4. *A Kafka Cluster with Ten Brokers*

# CPU Cores Per Broker	RAM Per Broker	Disk Storage Per Broker (NVMe)	%us	%sy	%wa
16	122GB	3.8TB (2 disks)	22%	12%	1%

Total CPU% Utilization	Normalized Load Average	Used Disk Space in /var/lib/kafka	Network Processor Idle
35%	0.4	75%	85%

Findings

CPU utilization: The cluster is over-provisioned in terms of CPU (only 35% CPU utilization).

Disk storage usage: The cluster uses 75 percent of its disk storage, which currently is enough.

Disk IOPS utilization: The `wa%` is 1 percent, which means there are almost no reads/writes to the disks.

Load on the cluster: The NLA is only 0.4, which isn't high.

Network processor idle: This is lower than expected, which shows there's either some bottleneck in the cluster or that there are not enough threads.

Options for Reducing Costs

Scale Down

To scale down the brokers, you'll need to use instances with eight cores, 16GB RAM, and one disk with 1.9TB storage. Let's see if these are enough:

- *CPU:* The cluster currently uses 35 percent of the CPU, so with half of the cores, the CPU utilization is expected to reach 70 percent, which is okay.
- *RAM:* There are very few reads from the disks, so all you can tell is that the current amount of RAM is sufficient for the brokers in order to avoid almost completely accessing the disks in order to read data that doesn't exist in the page cache. However, you can't tell whether with half of the current RAM, the brokers won't need to access the disks. The only way to know this is to scale down the cluster and monitor the CPU `wa%` (using the `top` command), the disk utilization (using the `iostat` command), and the misses from the page cache (using the `cachestat` script). These metrics will give you an indication whether the page cache has enough RAM in order to prevent the brokers from accessing the disks in order to serve the fetch requests of the consumers and other brokers (as part of the replication process).
- *Disk storage:* Since the current storage usage is already 75 percent, cutting the disk storage by half will leave the cluster with less storage capacity than required. So if the cluster will be scaled-down, you'll need to attach more disks to this broker, which is possible. In terms of disk storage and utilization, everything will remain the same since you'll keep the same amount of storage and IOPS as before.
- *Load on the cluster:* The normalized load average on the cluster is expected to be around 0.8, since the current load is 0.4 and you're going to reduce the cores by half, and it seems that most of the load originates from CPU us% utilization. Such a load is okay for Kafka clusters, but the cluster shouldn't reach a higher load than that.

Scale In

To scale in the cluster, you can remove five brokers so that you'll be left with five brokers, each with the same number of cores, RAM, disk storage, and IOPS. The previous arguments for CPU, RAM, disk storage, and load on the cluster apply here. In order to keep the same amount of disk storage, you'll need to attach two more disks per broker, which is also possible.

Recommendation

You can either scale down the cluster by half or scale in the cluster by half, but in both cases, you'll need to attach more disks in order to have the same amount of storage.

However, the issue of the low network processor idle% makes the decision whether to reduce resources from the cluster a tough one. This metric indicates the percentage of time that the network processor threads in a Kafka broker were idle and not processing any incoming requests from clients. A value of 85 percent means that for 15 percent of the time, the network threads were busy, and my experience shows that clients of Kafka clusters with such busy network threads usually experience some latency.

Although from the perspective of CPU, RAM utilization, and load on the cluster, it seems that you could reduce half of the cores, RAM, and hardware costs of the cluster, the low network threads idle% metric indicates there's a bottleneck that could cause clients of the clusters even greater latency.

That's why in the case of this cluster, it's better to check whether the clients suffer from latency rather than trying to scale down or scale in the cluster.

Example 5

Table 12-5 shows a Kafka cluster with eight brokers.

Table 12-5. *A Kafka Cluster with Eight Brokers*

# CPU Cores Per Broker	RAM Per Broker	Disk Storage Per Broker (NVMe)	%us	%sy	%wa
24	192GB	15TB (2 disks)	14%	5%	1%

Total CPU% Utilization	Normalized Load Average	Used Disk Space in /var/lib/kafka	Network Processor Idle
20%	0.25	40%	99%

Findings

CPU utilization: The cluster is over-provisioned in terms of CPU (only 20% CPU utilization).

Disk storage usage: The cluster uses 40 percent of its disk storage, which currently is enough.

Disk IOPS utilization: The `wa%` is 1 percent, which means there are almost no reads/writes to the disks.

Load on the cluster: The NLA is only 0.25, which isn't high.

Options for Reducing Costs

This is a pretty simple example of a cluster that can use brokers with half the cores, RAM, and storage. It can be achieved by either scaling down the cluster and using eight brokers with half the cores, RAM and storage, or by scaling in the cluster and using four brokers with the same instance type instead of eight brokers.

Example 6

Table 12-6 shows a Kafka cluster with six brokers.

Table 12-6. *A Kafka Cluster with Six Brokers*

# CPU Cores Per Broker	RAM Per Broker	Disk Storage Per Broker (NVMe)	%us	%sy	%wa
12	96GB	7.5TB	20%	7%	1%

Total CPU% Utilization	Normalized Load Average	Used Disk Space in /var/lib/kafka	Network Processor Idle
30%	0.35	50%	99%

Findings

CPU utilization: The cluster is over-provisioned in terms of CPU (only 30% CPU utilization).

Disk storage usage: The cluster uses only 50 percent of its disk storage.

Disk IOPS utilization: The `wa%` is 1 percent, which means there are almost no reads/writes to the disks.

Load on the cluster: The NLA is only 0.35, which isn't high.

Options for Reducing Costs

Scale Down

In AWS, you can use a lower instance type from the same instance family with eight cores, 64GB RAM, and 5TB storage (composed of two 2.5GB disks). This should increase the CPU utilization of the cluster to ±60 percent, the normalized load average to ±0.7, and the disk usage to 75 percent.

This move will reduce the cost of the cluster's hardware by 33 percent without reaching any CPU or storage bottleneck.

Scale In

By scaling down the cluster from 12 brokers to 8, you can anticipate the CPU usage to increase from 40 to 60 percent and storage usage to rise from 50 to 75 percent. This adjustment is projected to lead to a 33 percent reduction in hardware costs without hitting any CPU or storage constraints. To facilitate this change, it's essential to reassign the partitions across all topics in the cluster and adjust the number of partitions to ensure a balanced distribution among the remaining brokers.

Recommendation

Scaling in the cluster would reduce the hardware costs by 30 percent, since all the brokers will be replaced with brokers at 2/3 the price (and 2/3 the hardware resources). This will require migrating all the topics to the new brokers without having to change the number of partitions.

Scaling the cluster down would also reduce the costs by 33 percent and will require a reassignment of the partitions. It will also require a change in the number of partitions of all the topics in which their number of partitions doesn't divide equally by 8.

In this case, there's no better or worse approach, since both reduce the cost of the cluster equally.

Summary

Scaling and optimizing Kafka clusters is a multifaceted challenge that requires a comprehensive understanding of both technical intricacies and financial considerations. This chapter delved into the various strategies for handling different scaling requirements, focusing on both cloud-based and on-premises Kafka clusters, and emphasized the importance of avoiding over-provisioning to reduce costs.

We began by exploring the constraints on resources such as RAM, CPU cores, and disk storage, and provided potential solutions, weighing their respective merits and challenges. While the cloud-based environment often affords more flexibility with options to scale up or out, on-premises clusters call for a meticulous evaluation of existing hardware and careful alignment with financial and managerial objectives. The importance of optimizing cost savings without sacrificing performance was highlighted, underscoring the need for accurate evaluation of actual workloads.

Furthermore, we offered insights into relevant Kafka metrics and OS considerations for assessing cluster usage, and provided six examples of over-provisioned clusters, illustrating how scaling in and scaling down of these clusters can be implemented. These practical examples elucidated the optimal scaling options to maximize cost reduction while maintaining stability.

It's important to emphasize that scaling and cost reduction of Kafka clusters are not merely technical endeavors, but are strategic undertakings that necessitate thoughtful planning and a well-rounded understanding of various influencing factors.