

# [Kafka-users] Cluster design distribution and JBOD vs RAID



(/user/Bert-

Bert Corderman (/user/Bert-Corderman/vAFsBWwmMKfDpxLj37vK1b)

Corderman/vAFsBWwmMKfDpxLj37vK1b) Apr 16, 2014 at 5:58 pm (/g/kafka/users/2014/04)

I am wondering what others are doing in terms of cluster separation. (if at all) For example let's say I need 24 nodes to support a given workload.

What are the tradeoffs between a single 24 node cluster vs 2 x 12 node clusters for example. The application I support can support separation of data fairly easily as the data is all processed in the same way but can be sharded isolated based on customers. I understand the standard tradeoffs, for example putting all your eggs in one basket but curious as if there are any details specific to Kafka in terms of cluster scale out.

Somewhat related is the use of RAID vs JBOD, I have reviewed the documents on the Kafka site and understand the tradeoff between space as well as sequential IO vs random and the fact a RAID rebuild might kill the system.

I am specifically asking the question as it relates to larger cluster and the impact on the number of partitions a topic might need.

Take an example of a 24 node cluster with 12 drives each the cluster would have 288 drives. To ensure a topic is distributed across all drives a topic would require 288 partitions. I am planning to test some of this but wanted to know if there was a rule of thumb. The following link <https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdoIchoosethenumberofpartitionsfortopic?> (<https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdoIchoosethenumberofpartitionsfortopic?>) Talks about supporting up to 10K partitions but its not clear if this is for a cluster as a whole vs topic based

Those of you running larger clusters what are you doing?

Bert

reply ([mailto:users%40kafka.apache.org?In-Reply-](mailto:users%40kafka.apache.org?In-Reply-To=%3CCANL%3DHfSmetDtSkpcbqE9NVKoVVJG9Axw6uR6TPUGhGq%3Dr%2B3YA%40mail.gmail.com%3E&Sut)

To=%3CCANL%3DHfSmetDtSkpcbqE9NVKoVVJG9Axw6uR6TPUGhGq%3Dr%2B3YA%40mail.gmail.com%3E&Sut

Tweet (<https://twitter.com/share>)



## Search Discussions

☐ All Groups ☒ users

10 responses

## Oldest (/t/kafka/users/144g5k77qx/cluster-design-distribution-and-jbod-vs-raid#responses\_tab\_top)

### Nested (/t/kafka/users/144g5k77qx/cluster-design-distribution-and-jbod-vs-raid#responses\_tab\_top)

 (/user/Bello-**Bello, Bob** (/user/Bello-Bob/rJNCA74oM5owWU1Tu6r8tD) at **Apr 16, 2014**  
Bob/rJNCA74oM5owWU1Tu6r8tD) at **11:41 pm** 

Perhaps as you consider the size of your cluster, a few questions about the kind of messaging you are looking at? I can use an example of what we do in our production environment while not going into specifics. These are just observations from an OPS perspective. (sorry for the wall of text.)

\* Size of messages (<100 bytes, <1kB, <10kB, <100kB, <1MB, <10MB, etc). (we run messages size between a few bytes to over 100KB with a few at over 1MB).

\* Volume of messages per second (we produce over 15k per second and can consume over 100K per second when we are processing though some lag)

\* # of Producer clients (a few, a lot) (we have over 300 app servers the produce messages to the Kafka cluster)

\*\* Not only does this affect Kafka broker performance but it can use a lot of TCP connections specially if you run a large Kafka cluster

\* # of Consumer clients (a few, a lot) (we have less than 50 app servers that consume at this time)

\*\* This also affects the # of TCP connections to Kafka brokers. (We have over 2400+ TCP connections to our cluster)

\* Will you compress your message before sending them to Kafka? (we have a mix of snappy, gzip and non-compressed messages depending on the application). This can affect your disk usage

\* Planned retention period. Longer retention period = more storage required. (we have varied retention periods per topic, between 10 days and 30 days).

\* The number of topics per cluster. I believe Kafka scales well with the number of topics, however you have to worry about a few things:

\*\* More topics, means slower migration/failover when Kafka brokers are shutdown or fail. This has caused us time out issues. Planned shutdown of a Kafka broker can take over 30 seconds to over 3 minutes. (We have over >10 and <50 topics. We are growing topics rapidly.)

\* The number of partitions per topic. More partitions per topic = more open file handles, (2 per log file, one for data and one more the index). We run average of 130 partitions. You have to consider your cardinality for your messages if order is important. Can you use a key that allows a good distribution across partitions while maintaining order? If all your message end up in just a few partitions within the topic then it's harder scale the consumption. This all depends on your use case.

It might sound like good rationale to scale the # of partitions for a topic to a huge number (for just in case). I think it all depends.

\* How many consumer threads can consume a single topic? You can't go wider than the # of partitions however Kafka clients easily work with a large # of partitions with a few consumer threads.

\* Producer vs. Consumer size. Is your messaging flow Producer or Consumer heavy. Kafka is awesome and sending data to consumers that use "recent" data. Since Kafka uses memory mapped files, any data from Kafka that is in RAM will be very fast. (Our servers have 256GB of ram on them).

\* Size of your cluster vs. the # of replicas. Larger # of Kafka brokers means more chance of failure within the cluster. Same kind of reason why you generally won't see a large RAID5 array. You get one failure before you lose data. If you decide to run a large cluster and # of replicas will be important. How much risk are you willing to take? (We run a 6 node cluster with a replica factor of 3. We can lose a total of two nodes before losing data).

\* Are you running on native iron or virtualized? VM is generally lower performance but can generally spin up new instances faster upon failure. We run on native iron so we get excellent performance at the cost of longer lead times to provision new Kafka brokers.

\* Networking. Are you are running 100mbit, 1gig or 10gib? You can only produce and consume so much data. Larger clusters let you run a total aggregate bandwidth. Don't forget about replication! Topic/partition leaders must replicate to all replica Kafkabrokers (hub/spoke). How long can you wait for replication to occur after a planned or un-planned outage? (We run >1Gig).

\* Monitoring. Large # of Kafka brokers means more to monitor. Do you have a centralized monitoring app? Kafka provides a lot (huge!) JMX information. Making sense of it all can take some time.

\* Disk I/O. JBOD vs. RAID. How much are you willing to tolerate failures? Do you have provisioned IO? (We run native iron and local disk in a RAID configuration. It was easier for us to manage a single mount point than a bunch in a JBOD configuration. We rely on local RAID and Kafka replication to keep enough copies of our data. We have a large amount of disk capacity. We can tolerate large re-replication events due to broker failure without affecting producer or consumer performance.)

\* Disk capacity / Kafka Broker capacity. Depending on your volume, message size and retention period, how much disk space will you need? (Using our "crystal ball tech(tm)" we decided over 20TB per Kafka broker would meet our needs. We will probably add Kafka brokers over adding disk as we outgrow this.)

\* Separate clusters to keep information separated? Do you have a use case for keeping customer data separate? Compliance use cases such as PCI or SOX? This may be a good reason to keep separate Kafka clusters. I assume that you already will keep separate clusters for DEV/QA/PROD.

\* Zookeeper performance - 3 node, 5 node or 7 node. Less nodes, better performance. More nodes, better failure tolerance. We run 5 nodes with the transaction logs on SSD. Our ZK update performance is very good.

# of partitions per Topic debate:

Personally, I'm a proponent of larger # of partitions per topic without going way large. You can add Kafka Brokers to increase capacity and get more performance. However though it's possible to add partitions after a topic is created, it can cause issues with your key hashing depending on your message architecture.

\* Increasing # of brokers = easy

\* Increasing the # of partitions in a topic with data in it = hard

For us, we will be adding more topics and as we add additional messaging functionality.

Example:

130 partitions per topic / 6 brokers = 5 leader partitions per broker per topic. If you replicate 3 the you will end up with 3x active partitions per broker.

1024 partitions per topic / 24 brokers =~ 43 leader partitions per broker per topic.

Final thoughts:

There's no magical formula for this as already stated in the wiki. It is a lot of trial and error. I will say that we went from a few 100 messages per second volume to over 40k per second by adding one application and our Kafka cluster didn't even blink.

Kafka is awesome.

Btw, we're running 0.8.0.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert Corderman

Sent: Wednesday, April 16, 2014 11:58 AM

To: users@kafka.apache.org

Subject: Cluster design distribution and JBOD vs RAID

I am wondering what others are doing in terms of cluster separation. (if at all) For example let's say I need 24 nodes to support a given workload. What are the tradeoffs between a single 24 node cluster vs 2 x 12 node clusters for example. The application I support can support separation of data fairly easily as the data is all processed in the same way but can be sharded isolated based on customers. I understand the standard tradeoffs, for example putting all your eggs in one basket but curious as if there are any details specific to Kafka in terms of cluster scale out.

Somewhat related is the use of RAID vs JBOD, I have reviewed the documents on the Kafka site and understand the tradeoff between space as well as sequential IO vs random and the fact a RAID rebuild might kill the system. I am specifically asking the question as it relates to larger cluster and the impact on the number of partitions a topic might need.

Take an example of a 24 node cluster with 12 drives each the cluster would have 288 drives. To ensure a topic is distributed across all drives a topic would require 288 partitions. I am planning to test some of this but wanted to know if there was a rule of thumb. The following link <https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-Howdolchoosethenumberofpartitionsfortopic?> (<https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-Howdolchoosethenumberofpartitionsfortopic?>) Talks about supporting up to 10K partitions but its not clear if this is for a cluster as a whole vs topic based

Those of you running larger clusters what are you doing?

Bert

reply (mailto:users%40kafka.apache.org?In-Reply-To=%3C03EAA0700DF21749A1BCA520739FE43CC5873372%40MER2-EXCH07A3.echostar.com%3E&Subject=RE%3A%20Cluster%20design%20distribution%20and%20JBOD%20vs%20RAID) | permalink (/p/kafka/users/144gfcqv0/cluster-design-distribution-and-jbod-vs-raid)

**Bert Corderman (/user/Bert-Corderman/vAFsBWwmMKfDpxLj37vK)**

 (/user/Bert-Corderman/vAFsBWwmMKfDpxLj37vK1b)

Hey Bob,

thanks for your detailed response. I have added comments inline.

*On Wed, Apr 16, 2014 at 7:41 PM, Bello, Bob wrote:*

Perhaps as you consider the size of your cluster, a few questions about the kind of messaging you are looking at? I can use an example of what we do in our production environment while not going into specifics. These are just observations from an OPS perspective. (sorry for the wall of text.)

\* Size of messages (<100 bytes, <1kB, <10kB, <100kB, <1MB, <10MB, etc). (we run messages size between a few bytes to over 100KB with a few at over 1MB).

BERT> We have several use cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at use cases that have varying message sizes of 200, 300, 1000, 2200 bytes

\* Volume of messages per second (we produce over 15k per second and can consume over 100K per second when we are processing though some lag)

BERT> The use case we are looking at currently has hourly peaks of about 450K messages per second. For sizing we want to make sure we can support 900K. Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

\* # of Producer clients (a few, a lot) (we have over 300 app servers the produce messages to the Kafka cluster)

\*\* Not only does this affect Kafka broker performance but it can use a lot of TCP connections specially if you run a large Kafka cluster

BERT> our producer count will be low ...maybe 8-16 hosts.

\* # of Consumer clients (a few, a lot) (we have less than 50 app servers that consume at this time)

\*\* This also affects the # of TCP connections to Kafka brokers. (We have over 2400+ TCP connections to our cluster)

BERT> This will be much higher but not sure yet. We are also looking at replacing some legacy technology with storm so this is a bit up in the air right now.

\* Will you compress your message before sending them to Kafka? (we have a mix of snappy, gzip and non-compressed messages depending on the application). This can affect your disk usage

BERT> We will use whatever performs best ;) My gut is that we will be using snappy

\* Planned retention period. Longer retention period = more storage required. (we have varied retention periods per topic, between 10 days and 30 days).

\* The number of topics per cluster. I believe Kafka scales well with the number of topics, however you have to worry about a few things:

\*\* More topics, means slower migration/failover when Kafka brokers are shutdown or fail. This has caused us time out issues. Planned shutdown of a Kafka broker can take over 30 seconds to over 3 minutes. (We have over >1 and <50 topics. We are growing topics rapidly.)

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover and compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

\* The number of partitions per topic. More partitions per topic = more open file handles, (2 per log file, one for data and one more the index). We run average of 130 partitions. You have to consider your cardinality for your messages if order is important. Can you use a key that allows a good distribution across partitions while maintaining order? If all your message end up in just a few partitions within the topic then it's harder scale the consumption. This all depends on your use case.

BERT> We are lucky that order is not critical for our large feeds.

It might sound like good rationale to scale the # of partitions for a topic to a huge number (for just in case). I think it all depends.

\* How many consumer threads can consume a single topic? You can't go wider than the # of partitions however Kafka clients easily work with a large # of partitions with a few consumer threads.

\* Producer vs. Consumer size. Is your messaging flow Producer or Consumer heavy. Kafka is awesome and sending data to consumers that use "recent" data. Since Kafka uses memory mapped files, any data from Kafka that is in RAM will be very fast. (Our servers have 256GB of ram on them).

BERT> Our default config config has a 256GB of memory also. One thing I do want to test is impact on cluster of reading data not in memory. Have you done any testing like this?

\* Size of your cluster vs. the # of replicas. Larger # of Kafka brokers means more chance of failure within the cluster. Same kind of reason why you generally won't see a large RAID5 array. You get one failure before you lose data. If you decide to run a large cluster and # of replicas will be important. How much risk are you willing to take? (We run a 6 node cluster with a replica factor of 3. We can lose a total of two nodes before losing data).

BERT> Thanks for the datapoint. We were also planning to go with replication factor of 3

\* Are you running on native iron or virtualized? VM is generally lower performance but can generally spin up new instances faster upon failure. We run on native iron so we get excellent performance at the cost of longer lead times to provision new Kafka brokers.

BERT> We are big fans of vms...however kafka will be on physical

\* Networking. Are you are running 100mbit, 1gig or 10gib? You can only produce and consume so much data. Larger clusters let you run a total aggregate bandwidth. Don't forget about replication! Topic/partition leaders must replicate to all replica Kafkabrokers (hub/spoke). How long can you wait for replication to occur after a planned or un-planned outage? (We run >1Gig).

BERT> 10gb....so cheap now. I did cost analysis and found that a single 10gb port costs about the same as 2 x 1gig. Five times the bandwidth and less latency makes it no brainer. If your kafka hosts have multiple nics make sure they are using the right port. This one bit me for a little.  
(hostname config in the broker config)

\* Monitoring. Large # of Kafka brokers means more to monitor. Do you have a centralized monitoring app? Kafka provides a lot (huge!) JMX information. Making sense of it all can take some time.

BERT> We have not determined what to use just yet for monitoring. What are you guys using?

\* Disk I/O. JBOD vs. RAID. How much are you willing to tolerate failures? Do you have provisioned IO? (We run native iron and local disk in a RAID configuration. It was easier for us to manage a single mount point than a bunch in a JBOD configuration. We rely on local RAID and Kafka replication to keep enough copies of our data. We have a large amount of disk capacity. We can tolerate large re-replication events due to broker failure without affecting producer or consumer performance.)

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

\* Disk capacity / Kafka Broker capacity. Depending on your volume, message size and retention period, how much disk space will you need? (Using our "crystal ball tech(tm)" we decided over 20TB per Kafka broker would meet our needs. We will probably add Kafka brokers over adding disk as we outgrow this.)

BERT> I need a crystal ball ;)

\* Separate clusters to keep information separated? Do you have a use case for keeping customer data separate? Compliance use cases such as PCI or SOX? This may be a good reason to keep separate Kafka clusters. I assume that you already will keep separate clusters for DEV/QA/PROD.

BERT> yes DEV/QA/PROD completely separate

\* Zookeeper performance - 3 node, 5 node or 7 node. Less nodes, better performance. More nodes, better failure tolerance. We run 5 nodes with the transaction logs on SSD. Our ZK update performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

# of partitions per Topic debate:

Personally, I'm a proponent of larger # of partitions per topic without going way large. You can add Kafka Brokers to increase capacity and get more performance. However though it's possible to add partitions after a topic is created, it can cause issues with your key hashing depending on your message architecture.

\* Increasing # of brokers = easy

\* Increasing the # of partitions in a topic with data in it = hard

For us, we will be adding more topics and as we add additional messaging functionality.

Example:

130 partitions per topic / 6 brokers = 5 leader partitions per broker per topic. If you replicate 3 the you will end up with 3x active partitions per broker.

1024 partitions per topic / 24 brokers =~ 43 leader partitions per broker per topic.

BERT> Thanks for the example. Good to see others are using larger partition counts.

Final thoughts:

There's no magical formula for this as already stated in the wiki. It is a lot of trial and error. I will say that we went from a few 100 messages per second volume to over 40k per second by adding one application and our Kafka cluster didn't even blink.

Kafka is awesome.

Btw, we're running 0.8.0.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert Corderman

Sent: Wednesday, April 16, 2014 11:58 AM



To: users@kafka.apache.org

Subject: Cluster design distribution and JBOD vs RAID

I am wondering what others are doing in terms of cluster separation. (if at all) For example let's say I need 24 nodes to support a given workload.

What are the tradeoffs between a single 24 node cluster vs 2 x 12 node clusters for example. The application I support can support separation of data fairly easily as the data is all processed in the same way but can be sharded isolated based on customers. I understand the standard tradeoffs, for example putting all your eggs in one basket but curious as if there are any details specific to Kafka in terms of cluster scale out.

Somewhat related is the use of RAID vs JBOD, I have reviewed the documents on the Kafka site and understand the tradeoff between space as well as sequential IO vs random and the fact a RAID rebuild might kill the system.

I am specifically asking the question as it relates to larger cluster and the impact on the number of partitions a topic might need.

Take an example of a 24 node cluster with 12 drives each the cluster would have 288 drives. To ensure a topic is distributed across all drives a topic would require 288 partitions. I am planning to test some of this but wanted to know if there was a rule of thumb. The following link

<https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-Howdolchoosethenumberofpartitionsfortopic>  
(<https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-Howdolchoosethenumberofpartitionsfortopic>)  
?

Talks about supporting up to 10K partitions but its not clear if this is for a cluster as a whole vs topic based

Those of you running larger clusters what are you doing?

Bert

reply (mailto:users%40kafka.apache.org?In-Reply-

To=%3CCANL%3DHffEO25eAJJFEUD5h8PRbdvrhQWorTgAyjn8und\_M5\_VMA%40mail.gmail.com%3E&Subject=Re%3A%20cluster-design-distribution-and-jbod-vs-raid)  
| permalink (/p/kafka/users/144hym7a8g/cluster-design-distribution-and-jbod-vs-raid)



(/user/Bello-

**Bello, Bob (/user/Bello-Bob/rJNCA74oM5owWU1Tu6r8tD) at Apr 18,**

Bob/rJNCA74oM5owWU1Tu6r8tD)

**2014 at**

**12:00 am**



Some feedback from your feedback.

BERT> We have several uses cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at uses cases that have varying message sizes of 200, 300, 1000, 2200 bytes

BOB> Since you are using small message size, watch out or large index files. You can stuff a lot of messages in to the

default log file size of 512MB. We use 1GB log files before rolling them.

BERT> The use case we are looking at currently has hourly peaks of about 450K messages per second. For sizing we want to make sure we can support 900K. Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

BOB> I believe LinkedIn has reported getting a throughput of 900k messages though a 6 node cluster. If you can achieve a flush rate of 100MB/s (which is easy for a good RAID setup) having a 12 node cluster should be doable. Remember when your topic/partition leadership is balance across the cluster (preferred replica election) you get to take advantage of all the brokers. Don't forget to architect for a failures. Can your cluster handle max throughput with two Kafka broker in an offline state?

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover and compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

BOB> Yes, that is what we have seen. The current controller architecture takes longer for Kafka nodes to fail over. It's not the # of topics, but the # of topic/partitions that have to move over. When a Kafka broker fails (planned or unplanned), the producers and the consumers have to pause for all the topic/partition pairs that were the leader for the off line Kafka broker and they have to move to another Kafka broker that is in ISR. By having lots of topics/partitions (we have many thousands), it can take a bit. Remember it's only a chunk, not all topics and partitions. This of course can change as the Kafka development team changes how this works. I highly recommend creating your topic and partition counts in DEV/QA and test this out. You will see a difference.

As for the amount of data in the topic/partition that is of no concern for failover. The Kafka broker will only failover those topics/partitions that are in ISR. Replication time once a Kafka broker is brought back online will depend on how far behind the Kafka broker is from the leader. This is delta in the offset. Planned shutdowns can be minutes, unplanned shutdowns/failures can take hours for our data to re-replicate.

BERT> Our default config config has a 256GB of memory also. One thing I do want to test is impact on cluster of reading data not in memory. Have you done any testing like this?

BOB> Yes, it's about putting enough data to flush outside the OS file cache. But 512GB of data in your topics to make sure the data is not in the cache. Also, you can reset and/or use new consumer groups and make sure you read from the lowest-offset. Watch your iostats to see if you get lots of reads. On a normal Kafka cluster that is reading cached memory (for consumption), you will not see read IO. Assuming you don't have other processes on the system reading data (such as log aggregation). We see 30MB writes/flushes ever-other-second with 1-2% IO utilization.

BERT> We have not determined what to use just yet for monitoring. What are you guys using?

BOB> We are using a commercial APM solution. It's an java agent the plugs into the JVM on boot time. This reads the JMX information as well as file I/O rates, NIO rates and GC. It sends to a centralized monitoring console. Google "Java APM" for some ideas.

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

BOB> We are using RAID10. It was a requirement from our Unix guys. The rationale for this was we didn't want to lose just a disk and to have to rebuild/re-replicate 20TB of data. We haven't experienced any drive failures that I am aware of. We have had complete server failures, but the data was still good. I believe we have 10-4TB drives in a RAID10 configuration. I/O performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

BOB> We started with SSD. Kafka brokers itself doesn't write a lot of data frequently (to zookeeper). It's really about how your consumers flush their offsets. This is assuming you will be using the high-level consumer client. If you are going to flush the offsets to zookeeper on every message consumed (to get best effort nearly-exactly-once processing). You will be writing a lot of data to zookeeper. On our 5 node zookeeper cluster, we are doing 300+ writes per second, and can spike up to many 1000's. Typically it's 1-2MBs data rate. The SSDs are under 2% I/O utilization. 200MB of ZK data, and we clean up the files once per hour. We run some consumers in batch and flush on time delay. Other consumers are flush per message processed. It's the flush per message that causes the high-volume.

Push back on DEVs and software architecture if they want to flush per message. Do it where it's only absolutely necessary. :)

The high level Kafka consumer is good at "at least once" processing. Exactly once is a harder nut to crack. Exactly once processing may require some custom code around the low-level Kafka consumer client.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert Corderman  
Sent: Thursday, April 17, 2014 7:21 AM  
To: users@kafka.apache.org  
Subject: Re: Cluster design distribution and JBOD vs RAID

Hey Bob,

thanks for your detailed response. I have added comments inline.

*On Wed, Apr 16, 2014 at 7:41 PM, Bello, Bob wrote:*

Perhaps as you consider the size of your cluster, a few questions about the kind of messaging you are looking at? I can use an example of what we do in our production environment while not going into specifics. These are

just observations from an OPS perspective. (sorry for the wall of text.)

\* Size of messages (<100 bytes, <1kB, <10kB, <100kB, <1MB, <10MB, etc).  
(we run messages size between a few bytes to over 100KB with a few at over 1MB).

BERT> We have several use cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at use cases that have varying message sizes of 200, 300, 1000, 2200 bytes

\* Volume of messages per second (we produce over 15k per second and can consume over 100K per second when we are processing though some lag)

BERT> The use case we are looking at currently has hourly peaks of about 450K messages per second. For sizing we want to make sure we can support 900K. Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

\* # of Producer clients (a few, a lot) (we have over 300 app servers the produce messages to the Kafka cluster)

\*\* Not only does this affect Kafka broker performance but it can use a lot of TCP connections specially if you run a large Kafka cluster  
BERT> our producer count will be low ...maybe 8-16 hosts.

\* # of Consumer clients (a few, a lot) (we have less than 50 app servers that consume at this time)

\*\* This also affects the # of TCP connections to Kafka brokers. (We have over 2400+ TCP connections to our cluster)

BERT> This will be much higher but not sure yet. We are also looking at replacing some legacy technology with storm so this is a bit up in the air right now.

\* Will you compress your message before sending them to Kafka? (we have a mix of snappy, gzip and non-compressed messages depending on the application). This can affect your disk usage

BERT> We will use whatever performs best ;) My gut is that we will be using snappy

\* Planned retention period. Longer retention period = more storage required. (we have varied retention periods per topic, between 10 days and 30 days).

\* The number of topics per cluster. I believe Kafka scales well with the number of topics, however you have to worry about a few things:

\*\* More topics, means slower migration/failover when Kafka brokers are shutdown or fail. This has caused us time out issues. Planned shutdown of a Kafka broker can take over 30 seconds to over 3 minutes. (We have over >10 and <50 topics. We are growing topics rapidly.)

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover and compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

\* The number of partitions per topic. More partitions per topic = more open file handles, (2 per log file, one for data and one more the index). We run average of 130 partitions. You have to consider your cardinality for your messages if order is important. Can you use a key that allows a good distribution across partitions while maintaining order? If all your message end up in just a few partitions within the topic then it's harder scale the consumption. This all depends on your use case.

BERT> We are lucky that order is not critical for our large feeds.

It might sound like good rationale to scale the # of partitions for a topic to a huge number (for just in case). I think it all depends.

\* How many consumer threads can consume a single topic? You can't go wider than the # of partitions however Kafka clients easily work with a large # of partitions with a few consumer threads.

\* Producer vs. Consumer size. Is your messaging flow Producer or Consumer heavy. Kafka is awesome and sending data to consumers that use "recent" data. Since Kafka uses memory mapped files, any data from Kafka that is in RAM will be very fast. (Our servers have 256GB of ram on them).

BERT> Our default config config has a 256GB of memory also. One thing I do want to test is impact on cluster of reading data not in memory. Have you done any testing like this?

\* Size of your cluster vs. the # of replicas. Larger # of Kafka brokers means more chance of failure within the cluster. Same kind of reason why you generally won't see a large RAID5 array. You get one failure before you lose data. If you decide to run a large cluster and # of replicas will be important. How much risk are you willing to take? (We run a 6 node cluster with a replica factor of 3. We can lose a total of two nodes before losing data).

BERT> Thanks for the datapoint. We were also planning to go with replication factor of 3

\* Are you running on native iron or virtualized? VM is generally lower performance but can generally spin up new instances faster upon failure. We run on native iron so we get excellent performance at the cost of longer lead times to provision new Kafka brokers.

BERT> We are big fans of vms...however kafka will be on physical

\* Networking. Are you are running 100mbit, 1gig or 10gib? You can only produce and consume so much data. Larger clusters let you run a total aggregate bandwidth. Don't forget about replication! Topic/partition leaders must replicate to all replica Kafkabrokers (hub/spoke). How long can you wait for replication to occur after a planned or un-planned outage? (We run >1Gig).

BERT> 10gb....so cheap now. I did cost analysis and found that a single 10gb port costs about the same as 2 x 1gig. Five times the bandwidth and less latency makes it no brainer. If your kafka hosts have multiple nics make sure they are using the right port. This one bit me for a little.  
(hostname config in the broker config)

\* Monitoring. Large # of Kafka brokers means more to monitor. Do you have a centralized monitoring app? Kafka provides a lot (huge!) JMX information. Making sense of it all can take some time.

BERT> We have not determined what to use just yet for monitoring. What are you guys using?

\* Disk I/O. JBOD vs. RAID. How much are you willing to tolerate failures? Do you have provisioned IO? (We run native iron and local disk in a RAID configuration. It was easier for us to manage a single mount point than a bunch in a JBOD configuration. We rely on local RAID and Kafka replication to keep enough copies of our data. We have a large amount of disk capacity. We can tolerate large re-replication events due to broker failure without affecting producer or consumer performance.)

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

\* Disk capacity / Kafka Broker capacity. Depending on your volume, message size and retention period, how much disk space will you need? (Using our "crystal ball tech(tm)" we decided over 20TB per Kafka broker would meet our needs. We will probably add Kafka brokers over adding disk as we outgrow this.)

BERT> I need a crystal ball ;)

\* Separate clusters to keep information separated? Do you have a use case for keeping customer data separate? Compliance use cases such as PCI or SOX? This may be a good reason to keep separate Kafka clusters. I assume that you already will keep separate clusters for DEV/QA/PROD.

BERT> yes DEV/QA/PROD completely separate

\* Zookeeper performance - 3 node, 5 node or 7 node. Less nodes, better performance. More nodes, better failure tolerance. We run 5 nodes with the transaction logs on SSD. Our ZK update performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

# of partitions per Topic debate:

Personally, I'm a proponent of larger # of partitions per topic without going way large. You can add Kafka Brokers to increase capacity and get more performance. However though it's possible to add partitions after a topic is created, it can cause issues with your key hashing depending on your message architecture.

\* Increasing # of brokers = easy

\* Increasing the # of partitions in a topic with data in it = hard

For us, we will be adding more topics and as we add additional messaging functionality.

Example:

130 partitions per topic / 6 brokers = 5 leader partitions per broker per topic. If you replicate 3 the you will end up with 3x active partitions per broker.

1024 partitions per topic / 24 brokers =~ 43 leader partitions per broker per topic.

BERT> Thanks for the example. Good to see others are using larger partition counts.

Final thoughts:

There's no magical formula for this as already stated in the wiki. It is a lot of trial and error. I will say that we went from a few 100 messages per second volume to over 40k per second by adding one application and our Kafka cluster didn't even blink.

Kafka is awesome.

Btw, we're running 0.8.0.

- Bob

-----Original Message-----

From: bertoord@gmail.com On Behalf Of Bert

Corderman

Sent: Wednesday, April 16, 2014 11:58 AM

To: users@kafka.apache.org

Subject: Cluster design distribution and JBOD vs RAID

I am wondering what others are doing in terms of cluster separation. (if at all) For example let's say I need 24 nodes to support a given workload. What are the tradeoffs between a single 24 node cluster vs 2 x 12 node clusters for example. The application I support can support separation of data fairly easily as the data is all processed in the same way but can be sharded isolated based on customers. I understand the standard tradeoffs, for example putting all your eggs in one basket but curious as if there are any details specific to Kafka in terms of cluster scale out.

Somewhat related is the use of RAID vs JBOD, I have reviewed the documents on the Kafka site and understand the tradeoff between space as well as sequential IO vs random and the fact a RAID rebuild might kill the system. I am specifically asking the question as it relates to larger cluster and the impact on the number of partitions a topic might need.

Take an example of a 24 node cluster with 12 drives each the cluster would

have 288 drives. To ensure a topic is distributed across all drives a topic would require 288 partitions. I am planning to test some of this but wanted to know if there was a rule of thumb. The following link

<https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-Howdolchoosethenumberofpartitionsforatopic>  
(<https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-Howdolchoosethenumberofpartitionsforatopic>)  
?


Talks about supporting up to 10K partitions but its not clear if this is for a cluster as a whole vs topic based

Those of you running larger clusters what are you doing?

Bert

reply (mailto:users%40kafka.apache.org?In-Reply-To=%3C03EAA0700DF21749A1BCA520739FE43CC5873484%40MER2-EXCH07A3.echostar.com%3E&Subject=RE%3A%20Cluster%20design%20distribution%20and%20JBOD%20vs%20Raid%20performance) | permalink (/p/kafka/users/144j8nbxm0/cluster-design-distribution-and-jbod-vs-raid)

Andrew Otto (/user/Andrew-Otto/ANomU8wPFR6gnymSgRjQX9) at

 (/user/Andrew-Otto/ANomU8wPFR6gnymSgRjQX9)

At  
18  
20  
at  
2:  
pr

BOB> We are using RAID10. It was a requirement from our Unix guys. The rationale for this was we didn't want to lose just a disk and to have to rebuild/re-rotate 20TB of data. We haven't experienced any drive failures that I am aware of. We have had complete server failures, but the data was still good. I believe we have 10-4TB drives in a RAID10 configuration. I/O performance is very good.

Just curious, would losing one disk in a JBOD setup really mean you'd have to re-rotate 20TB of data? If a single drive dies, wouldn't you only lose the partitions that happen to be on that drive?

On Apr 17, 2014, at 8:00 PM, Bello, Bob wrote:

Some feedback from your feedback.

BERT> We have several use cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at use cases that have varying message sizes of 200, 300, 1000, 2200 bytes

BOB> Since you are using small message size, watch out for large index files. You can stuff a lot of messages in to the default log file size of 512MB. We use 1GB log files before rolling them.

BERT> The use case we are looking at currently has hourly peaks of about



450K messages per second. For sizing we want to make sure we can support 900K . Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

BOB> I believe LinkedIn has reported getting a throughput of 900k messages though a 6 node cluster. If you can achieve a flush rate of 100MB/s (which is easy for a good RAID setup) having a 12 node cluster should be doable. Remember when your topic/partition leadership is balance across the cluster (preferred replica election) you get to take advantage of all the brokers. Don't forget to architect for a failures. Can your cluster handle max throughput with two Kafka broker in an offline state?

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover and compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

BOB> Yes, that is what we have seen. The current controller architecture takes longer for Kafka nodes to fail over. It's not the # of topics, but the # of topic/partitions that have to move over. When a Kafka broker fails (planned or unplanned), the producers and the consumers have to pause for all the topic/partition pairs that were the leader for the off line Kafka broker and they have to move to another Kafka broker that is in ISR. By having lots of topics/partitions (we have many thousands), it can take a bit. Remember it's only a chunk, not all topics and partitions. This of course can change as the Kafka development team changes how this works. I highly recommend creating your topic and partition counts in DEV/QA and test this out. You will see a difference.

As for the amount of data in the topic/partition that is of no concern for failover. The Kafka broker will only failover those topics/partitions that are in ISR. Replication time once a Kafka broker is brought back online will depend on how far behind the Kafka broker is from the leader. This is delta in the offset. Planned shutdowns can be minutes, unplanned shutdowns/failures can take hours for our data to re-replicate.

BERT> Our default config has a 256GB of memory also. One thing I do want to test is impact on cluster of reading data not in memory. Have you done any testing like this?

BOB> Yes, it's about putting enough data to flush outside the OS file cache. But 512GB of data in your topics to make sure the data is not in the cache. Also, you can reset and/or use new consumer groups and make sure you read from the lowest-offset. Watch your iostats to see if you get lots of reads. On a normal Kafka cluster that is reading cached memory (for consumption), you will not see read IO. Assuming you don't have other processes on the system reading data (such as log aggregation). We see 30MB writes/flushes ever-other-second with 1-2% IO utilization.

BERT> We have not determined what to use just yet for monitoring. What are you guys using?

BOB> We are using a commercial APM solution. It's an java agent the plugs into the JVM on boot time. This reads the JMX information as well as file I/O rates, NIO rates and GC. It sends to a centralized monitoring console. Google "Java APM" for some ideas.

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard

server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

BOB> We are using RAID10. It was a requirement from our Unix guys. The rationale for this was we didn't want to lose just a disk and to have to rebuild/re-replicate 20TB of data. We haven't experienced any drive failures that I am aware of. We have had complete server failures, but the data was still good. I believe we have 10-4TB drives in a RAID10 configuration. I/O performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

BOB> We started with SSD. Kafka brokers itself doesn't write a lot of data frequently (to zookeeper). It's really about how your consumers flush their offsets. This is assuming you will be using the high-level consumer client. If you are going to flush the offsets to zookeeper on every message consumed (to get best effort nearly-exactly-once processing). You will be writing a lot of data to zookeeper. On our 5 node zookeeper cluster, we are doing 300+ writes per second, and can spike up to many 1000's. Typically it's 1-2MBs data rate. The SSDs are under 2% I/O utilization. 200MB of ZK data, and we clean up the files once per hour. We run some consumers in batch and flush on time delay. Other consumers are flush per message processed. It's the flush per message that causes the high-volume.

Push back on DEVs and software architecture if they want to flush per message. Do it where it's only absolutely necessary. :)

The high level Kafka consumer is good at "at least once" processing. Exactly once is a harder nut to crack. Exactly once processing may require some custom code around the low-level Kafka consumer client.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert Corderman

Sent: Thursday, April 17, 2014 7:21 AM

To: users@kafka.apache.org

Subject: Re: Cluster design distribution and JBOD vs RAID

Hey Bob,

thanks for your detailed response. I have added comments inline.

*On Wed, Apr 16, 2014 at 7:41 PM, Bello, Bob wrote:*

Perhaps as you consider the size of your cluster, a few questions about the kind of messaging you are looking at? I can use an example of what we do in our production environment while not going into specifics. These are just observations from an OPS perspective. (sorry for the wall of text.)

\* Size of messages (<100 bytes, <1kB, <10kB, <100kB, <1MB, <10MB, etc).

(we run messages size between a few bytes to over 100KB with a few at over 1MB).

BERT> We have several uses cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at uses cases that have varying message sizes of 200, 300, 1000, 2200 bytes

\* Volume of messages per second (we produce over 15k per second and can consume over 100K per second when we are processing though some lag)

BERT> The use case we are looking at currently has hourly peaks of about 450K messages per second. For sizing we want to make sure we can support 900K . Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

\* # of Producer clients (a few, a lot) (we have over 300 app servers the produce messages to the Kafka cluster)

\*\* Not only does this affect Kafka broker performance but it can use a lot of TCP connections specially if you run a large Kafka cluster

BERT> our producer count will be low ...maybe 8-16 hosts.

\* # of Consumer clients (a few, a lot) (we have less than 50 app servers that consume at this time)

\*\* This also affects the # of TCP connections to Kafka brokers. (We have over 2400+ TCP connections to our cluster)

BERT> This will be much higher but not sure yet. We are also looking at replacing some legacy technology with storm so this is a bit up in the air right now.

\* Will you compress your message before sending them to Kafka? (we have a mix of snappy, gzip and non-compressed messages depending on the application). This can affect your disk usage

BERT> We will use whatever performs best ;) My gut is that we will be using snappy

\* Planned retention period. Longer retention period = more storage required. (we have varied retention periods per topic, between 10 days and 30 days).

\* The number of topics per cluster. I believe Kafka scales well with the number of topics, however you have to worry about a few things:

\*\* More topics, means slower migration/failover when Kafka brokers are shutdown or fail. This has caused us time out issues. Planned shutdown of a Kafka broker can take over 30 seconds to over 3 minutes. (We have over >10 and <50 topics. We are growing topics rapidly.)

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover adn compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

\* The number of partitions per topic. More partitions per topic = more open file handles, (2 per log file, one for data and one more the index). We run average of 130 partitions. You have to consider your cardinality for your messages if order is important. Can you use a key that allows a good distribution across partitions while maintaining order? If all your message end up in just a few partitions within the topic then it's harder scale the consumption. This all depends on your use case.

BERT> We are lucky that order is not critical for our large feeds.

It might sound like good rationale to scale the # of partitions for a

topic to a huge number (for just in case). I think it all depends.

\* How many consumer threads can consume a single topic? You can't go wider than the # of partitions however Kafka clients easily work with a large # of partitions with a few consumer threads.

\* Producer vs. Consumer size. Is your messaging flow Producer or Consumer heavy. Kafka is awesome and sending data to consumers that use "recent" data. Since Kafka uses memory mapped files, any data from Kafka that is in RAM will be very fast. (Our servers have 256GB of ram on them).

BERT> Our default config config has a 256GB of memory also. One thing I do want to test is impact on cluster of reading data not in memory. Have you done any testing like this?

\* Size of your cluster vs. the # of replicas. Larger # of Kafka brokers means more chance of failure within the cluster. Same kind of reason why you generally won't see a large RAID5 array. You get one failure before you lose data. If you decide to run a large cluster and # of replicas will be important. How much risk are you willing to take? (We run a 6 node cluster with a replica factor of 3. We can lose a total of two nodes before losing data).

BERT> Thanks for the datapoint. We were also planning to go with replication factor of 3

\* Are you running on native iron or virtualized? VM is generally lower performance but can generally spin up new instances faster upon failure. We run on native iron so we get excellent performance at the cost of longer lead times to provision new Kafka brokers.

BERT> We are big fans of vms...however kafka will be on physical

\* Networking. Are you are running 100mbit, 1gig or 10gib? You can only produce and consume so much data. Larger clusters let you run a total aggregate bandwidth. Don't forget about replication! Topic/partition leaders must replicate to all replica Kafkabrokers (hub/spoke). How long can you wait for replication to occur after a planned or un-planned outage? (We run >1Gig).

BERT> 10gb....so cheap now. I did cost analysis and found that a single 10gb port costs about the same as 2 x 1gig. Five times the bandwidth and less latency makes it no brainer. If your kafka hosts have multiple nics make sure they are using the right port. This one bit me for a little. (hostname config in the broker config)

\* Monitoring. Large # of Kafka brokers means more to monitor. Do you have a centralized monitoring app? Kafka provides a lot (huge!) JMX information. Making sense of it all can take some time.

BERT> We have not determined what to use just yet for monitoring. What are you guys using?

\* Disk I/O. JBOD vs. RAID. How much are you willing to tolerate failures? Do you have provisioned IO? (We run native iron and local disk in a RAID configuration. It was easier for us to manage a single mount point than a bunch in a JBOD configuration. We rely of local RAID and Kafka replication to keep enough copies of our data. We have a large amount of disk capacity. We can tolerate large re-replication events due to broker failure without affecting producer or consumer performance.)

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID

rebuild and if so did it negatively impact the cluster. The standard server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

\* Disk capacity / Kafka Broker capacity. Depending on your volume, message size and retention period, how much disk space will you need? (Using our "crystal ball tech(tm)" we decided over 20TB per Kafka broker would meet our needs. We will probably add Kafka brokers over adding disk as we outgrow this.)

BERT> I need a crystal ball ;)

\* Separate clusters to keep information separated? Do you have a use case for keeping customer data separate? Compliance use cases such as PCI or SOX? This may be a good reason to keep separate Kafka clusters. I assume that you already will keep separate clusters for DEV/QA/PROD.

BERT> yes DEV/QA/PROD completely separate

\* Zookeeper performance - 3 node, 5 node or 7 node. Less nodes, better performance. More nodes, better failure tolerance. We run 5 nodes with the transaction logs on SSD. Our ZK update performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

# of partitions per Topic debate:

Personally, I'm a proponent of larger # of partitions per topic without going way large. You can add Kafka Brokers to increase capacity and get more performance. However though it's possible to add partitions after a topic is created, it can cause issues with your key hashing depending on your message architecture.

\* Increasing # of brokers = easy

\* Increasing the # of partitions in a topic with data in it = hard

For us, we will be adding more topics and as we add additional messaging functionality.

Example:

130 partitions per topic / 6 brokers = 5 leader partitions per broker per topic. If you replicate 3 the you will end up with 3x active partitions per broker.

1024 partitions per topic / 24 brokers =~ 43 leader partitions per broker per topic.

BERT> Thanks for the example. Good to see others are using larger partition counts.

Final thoughts:

There's no magical formula for this as already stated in the wiki. It is a lot of trial and error. I will say that we went from a few 100 messages per second volume to over 40k per second by adding one application and our

Kafka cluster didn't even blink.

Kafka is awesome.

Btw, we're running 0.8.0.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert

Corderman

Sent: Wednesday, April 16, 2014 11:58 AM

To: users@kafka.apache.org

Subject: Cluster design distribution and JBOD vs RAID

I am wondering what others are doing in terms of cluster separation. (if at all) For example let's say I need 24 nodes to support a given workload. What are the tradeoffs between a single 24 node cluster vs 2 x 12 node clusters for example. The application I support can support separation of data fairly easily as the data is all processed in the same way but can be sharded isolated based on customers. I understand the standard tradeoffs, for example putting all your eggs in one basket but curious as if there are any details specific to Kafka in terms of cluster scale out.

Somewhat related is the use of RAID vs JBOD, I have reviewed the documents on the Kafka site and understand the tradeoff between space as well as sequential IO vs random and the fact a RAID rebuild might kill the system. I am specifically asking the question as it relates to larger cluster and the impact on the number of partitions a topic might need.

Take an example of a 24 node cluster with 12 drives each the cluster would have 288 drives. To ensure a topic is distributed across all drives a topic would require 288 partitions. I am planning to test some of this but wanted to know if there was a rule of thumb. The following link

<https://wiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdIchoose thenumberofpartitionsforatopic>  
(<https://wiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdIchoose thenumberofpartitionsforatopic>)  
?


Talks about supporting up to 10K partitions but its not clear if this is for a cluster as a whole vs topic based

Those of you running larger clusters what are you doing?

Bert

reply (mailto:users%40kafka.apache.org?In-Reply-To=%3CDE6A1AE6-3D8F-4CCA-9222-58E2827FD4F2%40wikimedia.org%3E&Subject=Re%3A%20Cluster%20design%20distribution%20and%20JBOD%20vs%20raid) | permalink (/p/kafka/users/144jfxqc9e/cluster-design-distribution-and-jbod-vs-raid)

**Jay Kreps (/user/Jay-Kreps/1mSSJpzibcbeZQNGAa2WPO) at Apr 18, 2014 at 5:06 pm**

 (/user/Jay-Kreps/1mSSJpzibcbeZQNGAa2WPO)

If you lose one drive in a JBOD setup you will just re-replicate the data on that disk. It is similar to what you would do during RAID repair except that instead of having the data coming 100% from the mirror drives the load will be spread over the rest of the cluster.

The real downside of RAID is that if you lose one drive the machine with the dead drive will be down until the drive is replaced/fixed. RAID allows you to continue using that machine even with the dead drive, although we have seen that there can be issues with this in practice.

-Jay

*On Fri, Apr 18, 2014 at 7:35 AM, Andrew Otto wrote:*

BOB> We are using RAID10. It was a requirement from our Unix guys. The rationale for this was we didn't want to lose just a disk and to have to rebuild/re-replicate 20TB of data. We haven't experienced any drive failures that I am aware of. We have had complete server failures, but the data was still good. I believe we have 10-4TB drives in a RAID10 configuration. I/O performance is very good.

Just curious, would losing one disk in a JBOD setup really mean you'd have to re-replicate 20TB of data? If a single drive dies, wouldn't you only lose the partitions that happen to be on that drive?

*On Apr 17, 2014, at 8:00 PM, Bello, Bob wrote:*

Some feedback from your feedback.

BERT> We have several use cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at use cases that have varying message sizes of 200, 300, 1000, 2200 bytes

BOB> Since you are using small message size, watch out for large index files. You can stuff a lot of messages in to the default log file size of 512MB. We use 1GB log files before rolling them.

BERT> The use case we are looking at currently has hourly peaks of about

450K messages per second. For sizing we want to make sure we can support 900K . Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

BOB> I believe LinkedIn has reported getting a throughput of 900k messages though a 6 node cluster. If you can achieve a flush rate of 100MB/s (which is easy for a good RAID setup) having a 12 node cluster should be doable. Remember when your topic/partition leadership is balance across the cluster (preferred replica election) you get to take advantage of all the brokers. Don't forget to architect for a failures. Can your cluster handle max throughput with two Kafka broker in an offline state?

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover and compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

BOB> Yes, that is what we have seen. The current controller architecture takes longer for Kafka nodes to fail over. It's not the # of topics, but the # of topic/partitions that have to move over. When a Kafka broker fails (planned or unplanned), the producers and the consumers have to pause for all the topic/partition pairs that were the leader for the off line Kafka broker and they have to move to another Kafka broker that is in ISR. By having lots of topics/partitions (we have many thousands), it can take a bit. Remember it's only a chunk, not all topics and partitions. This of course can change as the Kafka development team changes how this works. I highly recommend creating your topic and partition counts in DEV/QA and test this out. You will see a difference.

As for the amount of data in the topic/partition that is of no concern for failover. The Kafka broker will only failover those topics/partitions that are in ISR. Replication time once a Kafka broker is brought back online will depend on how far behind the Kafka broker is from the leader. This is delta in the offset. Planned shutdowns can be minutes, unplanned shutdowns/failures can take hours for our data to re-replicate.

BERT> Our default config config has a 256GB of memory also. One thing I do want to test is impact on cluster of reading data not in memory. Have you done any testing like this?

BOB> Yes, it's about putting enough data to flush outside the OS file cache. But 512GB of data in your topics to make sure the data is not in the cache. Also, you can reset and/or use new consumer groups and make sure you read from the lowest-offset. Watch your iostats to see if you get lots of reads. On a normal Kafka cluster that is reading cached memory (for consumption), you will not see read IO. Assuming you don't have other processes on the system reading data (such as log aggregation). We see 30MB writes/flushes ever-other-second with 1-2% IO utilization.

BERT> We have not determined what to use just yet for monitoring. What are you guys using?

BOB> We are using a commercial APM solution. It's an java agent the



plugs into the JVM on boot time. This reads the JMX information as well as file I/O rates, NIO rates and GC. It sends to a centralized monitoring console. Google "Java APM" for some ideas.

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

BOB> We are using RAID10. It was a requirement from our Unix guys. The rationale for this was we didn't want to lose just a disk and to have to rebuild/re-replicate 20TB of data. We haven't experienced any drive failures that I am aware of. We have had complete server failures, but the data was still good. I believe we have 10-4TB drives in a RAID10 configuration. I/O performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

BOB> We started with SSD. Kafka brokers itself doesn't write a lot of data frequently (to zookeeper). It's really about how your consumers flush their offsets. This is assuming you will be using the high-level consumer client. If you are going to flush the offsets to zookeeper on every message consumed (to get best effort nearly-exactly-once processing). You will be writing a lot of data to zookeeper. On our 5 node zookeeper cluster, we are doing 300+ writes per second, and can spike up to many 1000's. Typically it's 1-2MBs data rate. The SSDs are under 2% I/O utilization. 200MB of ZK data, and we clean up the files once per hour. We run some consumers in batch and flush on time delay. Other consumers are flush per message processed. It's the flush per message that causes the high-volume.

Push back on DEVs and software architecture if they want to flush per message. Do it where it's only absolutely necessary. :)

The high level Kafka consumer is good at "at least once" processing. Exactly once is a harder nut to crack. Exactly once processing may require some custom code around the low-level Kafka consumer client.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert Corderman

Sent: Thursday, April 17, 2014 7:21 AM

To: users@kafka.apache.org

Subject: Re: Cluster design distribution and JBOD vs RAID

Hey Bob,

thanks for your detailed response. I have added comments inline.

*On Wed, Apr 16, 2014 at 7:41 PM, Bello, Bob wrote:*

Perhaps as you consider the size of your cluster, a few questions about the kind of messaging you are looking at? I can use an example of what we

do in our production environment while not going into specifics. These are

just observations from an OPS perspective. (sorry for the wall of text.)

- \* Size of messages (<100 bytes, <1kB, <10kB, <100kB, <1MB, <10MB, etc).

(we run messages size between a few bytes to over 100KB with a few at over

1MB).

BERT> We have several uses cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at uses cases that have varying message sizes of 200, 300, 1000, 2200 bytes

- \* Volume of messages per second (we produce over 15k per second and can consume over 100K per second when we are processing though some lag)

BERT> The use case we are looking at currently has hourly peaks of about 450K messages per second. For sizing we want to make sure we can support 900K . Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

- \* # of Producer clients (a few, a lot) (we have over 300 app servers the produce messages to the Kafka cluster)

\*\* Not only does this affect Kafka broker performance but it can use a lot

of TCP connections specially if you run a large Kafka cluster

BERT> our producer count will be low ...maybe 8-16 hosts.

- \* # of Consumer clients (a few, a lot) (we have less than 50 app servers that consume at this time)

\*\* This also affects the # of TCP connections to Kafka brokers. (We have over 2400+ TCP connections to our cluster)

BERT> This will be much higher but not sure yet. We are also looking at replacing some legacy technology with storm so this is a bit up in the air right now.

- \* Will you compress your message before sending them to Kafka? (we have a

mix of snappy, gzip and non-compressed messages depending on the application). This can affect your disk usage

BERT> We will use whatever performs best ;) My gut is that we will be using snappy

- \* Planned retention period. Longer retention period = more storage required. (we have varied retention periods per topic, between 10 days

and

30 days).

- \* The number of topics per cluster. I believe Kafka scales well with the number of topics, however you have to worry about a few things:

- \*\* More topics, means slower migration/failover when Kafka brokers are shutdown or fail. This has caused us time out issues. Planned shutdown

of a

Kafka broker can take over 30 seconds to over 3 minutes. (We have over 10 and <50 topics. We are growing topics rapidly.)

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover and compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

\* The number of partitions per topic. More partitions per topic = more open file handles, (2 per log file, one for data and one more the index).

We run average of 130 partitions. You have to consider your cardinality for

your messages if order is important. Can you use a key that allows a good

distribution across partitions while maintaining order? If all your message

end up in just a few partitions within the topic then it's harder scale the

consumption. This all depends on your use case.

BERT> We are lucky that order is not critical for our large feeds. It might sound like good rationale to scale the # of partitions for a topic to a huge number (for just in case). I think it all depends.

\* How many consumer threads can consume a single topic? You can't go wider

than the # of partitions however Kafka clients easily work with a large #

of partitions with a few consumer threads.

\* Producer vs. Consumer size. Is your messaging flow Producer or Consumer

heavy. Kafka is awesome and sending data to consumers that use "recent" data. Since Kafka uses memory mapped files, any data from Kafka that is in

RAM will be very fast. (Our servers have 256GB of ram on them).

BERT> Our default config config has a 256GB of memory also. One thing I do want to test is impact on cluster of reading data not in memory. Have you done any testing like this?

\* Size of your cluster vs. the # of replicas. Larger # of Kafka brokers means more chance of failure within the cluster. Same kind of reason why you generally won't see a large RAID5 array. You get one failure before you

lose data. If you decide to run a large cluster and # of replicas will be

important. How much risk are you willing to take? (We run a 6 node cluster

with a replica factor of 3. We can lose a total of two nodes before losing

data).

BERT> Thanks for the datapoint. We were also planning to go with replication factor of 3

\* Are you running on native iron or virtualized? VM is generally lower performance but can generally spin up new instances faster upon failure. We

run on native iron so we get excellent performance at the cost of longer lead times to provision new Kafka brokers.

BERT> We are big fans of vms...however kafka will be on physical

\* Networking. Are you are running 100mbit, 1gig or 10gib? You can only produce and consume so much data. Larger clusters let you run a total aggregate bandwidth. Don't forget about replication! Topic/partition leaders must replicate to all replica Kafkabrokers (hub/spoke). How long can you wait for replication to occur after a planned or un-planned outage?

(We run >1Gig).

BERT> 10gb.....so cheap now. I did cost analysis and found that a single 10gb port costs about the same as 2 x 1gig. Five times the bandwidth and less latency makes it no brainer. If your kafka hosts have multiple nics make sure they are using the right port. This one bit me for a little. (hostname config in the broker config)

\* Monitoring. Large # of Kafka brokers means more to monitor. Do you have

a centralized monitoring app? Kafka provides a lot (huge!) JMX information.

Making sense of it all can take some time.

BERT> We have not determined what to use just yet for monitoring. What are you guys using?

\* Disk I/O. JBOD vs. RAID. How much are you willing to tolerate failures?

Do you have provisioned IO? (We run native iron and local disk in a RAID configuration. It was easier for us to manage a single mount point than a

bunch in a JBOD configuration. We rely of local RAID and Kafka replication

to keep enough copies of our data. We have a large amount of disk capacity.

We can tolerate large re-replication events due to broker failure without

affecting producer or consumer performance.)

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

\* Disk capacity / Kafka Broker capacity. Depending on your volume, message

size and retention period, how much disk space will you need? (Using our "crystal ball tech(tm)" we decided over 20TB per Kafka broker would meet our needs. We will probably add Kafka brokers over adding disk as we outgrow this.)

BERT> I need a crystal ball ;)

\* Separate clusters to keep information separated? Do you have a use case

for keeping customer data separate? Compliance use cases such as PCI or SOX? This may be a good reason to keep separate Kafka clusters. I assume that you already will keep separate clusters for DEV/QA/PROD.

BERT> yes DEV/QA/PROD completely separate

\* Zookeeper performance - 3 node, 5 node or 7 node. Less nodes, better performance. More nodes, better failure tolerance. We run 5 nodes with the

transaction logs on SSD. Our ZK update performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

# of partitions per Topic debate:

Personally, I'm a proponent of larger # of partitions per topic without going way large. You can add Kafka Brokers to increase capacity and get more performance. However though it's possible to add partitions after a topic is created, it can cause issues with your key hashing depending on your message architecture.

\* Increasing # of brokers = easy

\* Increasing the # of partitions in a topic with data in it = hard

For us, we will be adding more topics and as we add additional messaging functionality.

Example:

130 partitions per topic / 6 brokers = 5 leader partitions per broker per

topic. If you replicate 3 the you will end up with 3x active partitions per broker.

1024 partitions per topic / 24 brokers =~ 43 leader partitions per broker

per topic.

BERT> Thanks for the example. Good to see others are using larger partition counts.

Final thoughts:

There's no magical formula for this as already stated in the wiki. It is a

lot of trial and error. I will say that we went from a few 100 messages per

second volume to over 40k per second by adding one application and our Kafka cluster didn't even blink.

Kafka is awesome.

Btw, we're running 0.8.0.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert

Corderman

Sent: Wednesday, April 16, 2014 11:58 AM

To: users@kafka.apache.org

Subject: Cluster design distribution and JBOD vs RAID

I am wondering what others are doing in terms of cluster separation.

(if at

all) For example let's say I need 24 nodes to support a given workload.

What are the tradeoffs between a single 24 node cluster vs 2 x 12 node

clusters for example. The application I support can support separation

of

data fairly easily as the data is all processed in the same way but can

be

sharded isolated based on customers. I understand the standard

tradeoffs,

for example putting all your eggs in one basket but curious as if there

are

any details specific to Kafka in terms of cluster scale out.

Somewhat related is the use of RAID vs JBOD, I have reviewed the documents

on the Kafka site and understand the tradeoff between space as well as

sequential IO vs random and the fact a RAID rebuild might kill the system.

I am specifically asking the question as it relates to larger cluster

and

the impact on the number of partitions a topic might need.

Take an example of a 24 node cluster with 12 drives each the cluster would

have 288 drives. To ensure a topic is distributed across all drives a

topic would require 288 partitions. I am planning to test some of this but

wanted to know if there was a rule of thumb. The following link

<https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-Howdolchoosethenumberofpartitionsfortopic>

([https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-](https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-Howdolchoosethenumberofpartitionsfortopic)

[Howdolchoosethenumberofpartitionsfortopic](https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-Howdolchoosethenumberofpartitionsfortopic))

?

Talks about supporting up to 10K partitions but its not clear if this is

for a cluster as a whole vs topic based

Those of you running larger clusters what are you doing?

Bert

reply (mailto:users%40kafka.apache.org?In-Reply-

To=%3CCA0eJiJj\_H\_oRfn%3DUZ6EUDx7UngBZSH4e3srcQWQrVGv3ukAOPA%40mail.gmail.com%3E&Subject=  
| permalink (/p/kafka/users/144j3epn1c/cluster-design-distribution-and-jbod-vs-raid)

**Bello, Bob (/user/Bello-Bob/rJNCA74oM5owWU1Tu6r8tD) at Apr**



(/user/Bello-

Bob/rJNCA74oM5owWU1Tu6r8tD)

**18,  
2014 at 7:01  
pm**

Yes you would lose the topic/partitions on the drive. I'm not quite sure if Kafka can determine what topics/partitions are missing or not. I suggest you try testing it.

- Bob

-----Original Message-----

From: Andrew Otto

Sent: Friday, April 18, 2014 8:36 AM

To: users@kafka.apache.org

Subject: Re: Cluster design distribution and JBOD vs RAID

BOB> We are using RAID10. It was a requirement from our Unix guys. The rationale for this was we didn't want to lose just a disk and to have to rebuild/re-replicate 20TB of data. We haven't experienced any drive failures that I am aware of. We have had complete server failures, but the data was still good. I believe we have 10-4TB drives in a RAID10 configuration. I/O performance is very good.

Just curious, would losing one disk in a JBOD setup really mean you'd have to re-replicate 20TB of data? If a single drive dies, wouldn't you only lose the partitions that happen to be on that drive?

*On Apr 17, 2014, at 8:00 PM, Bello, Bob wrote:*

Some feedback from your feedback.

BERT> We have several uses cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at uses cases that have varying message sizes of 200, 300, 1000, 2200 bytes

BOB> Since you are using small message size, watch out or large index files. You can stuff a lot of messages in to the default log file size of 512MB. We use 1GB log files before rolling them.

BERT> The use case we are looking at currently has hourly peaks of about 450K messages per second. For sizing we want to make sure we can support

900K . Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

BOB> I believe LinkedIn has reported getting a throughput of 900k messages though a 6 node cluster. If you can achieve a flush rate of 100MB/s (which is easy for a good RAID setup) having a 12 node cluster should be doable. Remember when your topic/partition leadership is balance across the cluster (preferred replica election) you get to take advantage of all the brokers. Don't forget to architect for a failures. Can your cluster handle max throughput with two Kafka broker in an offline state?

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover and compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

BOB> Yes, that is what we have seen. The current controller architecture takes longer for Kafka nodes to fail over. It's not the # of topics, but the # of topic/partitions that have to move over. When a Kafka broker fails (planned or unplanned), the producers and the consumers have to pause for all the topic/partition pairs that were the leader for the off line Kafka broker and they have to move to another Kafka broker that is in ISR. By having lots of topics/partitions (we have many thousands), it can take a bit. Remember it's only a chunk, not all topics and partitions. This of course can change as the Kafka development team changes how this works. I highly recommend creating your topic and partition counts in DEV/QA and test this out. You will see a difference.

As for the amount of data in the topic/partition that is of no concern for failover. The Kafka broker will only failover those topics/partitions that are in ISR. Replication time once a Kafka broker is brought back online will depend on how far behind the Kafka broker is from the leader. This is delta in the offset. Planned shutdowns can be minutes, unplanned shutdowns/failures can take hours for our data to re-replicate.

BERT> Our default config config has a 256GB of memory also. One thing I do want to test is impact on cluster of reading data not in memory. Have you done any testing like this?

BOB> Yes, it's about putting enough data to flush outside the OS file cache. But 512GB of data in your topics to make sure the data is not in the cache. Also, you can reset and/or use new consumer groups and make sure you read from the lowest-offset. Watch your iostats to see if you get lots of reads. On a normal Kafka cluster that is reading cached memory (for consumption), you will not see read IO. Assuming you don't have other processes on the system reading data (such as log aggregation). We see 30MB writes/flushes ever-other-second with 1-2% IO utilization.

BERT> We have not determined what to use just yet for monitoring. What are you guys using?

BOB> We are using a commercial APM solution. It's an java agent the plugs into the JVM on boot time. This reads the JMX information as well as file I/O rates, NIO rates and GC. It sends to a centralized monitoring console. Google "Java APM" for some ideas.

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard



server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

BOB> We are using RAID10. It was a requirement from our Unix guys. The rationale for this was we didn't want to lose just a disk and to have to rebuild/re-replicate 20TB of data. We haven't experienced any drive failures that I am aware of. We have had complete server failures, but the data was still good. I believe we have 10-4TB drives in a RAID10 configuration. I/O performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

BOB> We started with SSD. Kafka brokers itself doesn't write a lot of data frequently (to zookeeper). It's really about how your consumers flush their offsets. This is assuming you will be using the high-level consumer client. If you are going to flush the offsets to zookeeper on every message consumed (to get best effort nearly-exactly-once processing). You will be writing a lot of data to zookeeper. On our 5 node zookeeper cluster, we are doing 300+ writes per second, and can spike up to many 1000's. Typically it's 1-2MBs data rate. The SSDs are under 2% I/O utilization. 200MB of ZK data, and we clean up the files once per hour. We run some consumers in batch and flush on time delay. Other consumers are flush per message processed. It's the flush per message that causes the high-volume.

Push back on DEVs and software architecture if they want to flush per message. Do it where it's only absolutely necessary. :)

The high level Kafka consumer is good at "at least once" processing. Exactly once is a harder nut to crack. Exactly once processing may require some custom code around the low-level Kafka consumer client.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert Corderman

Sent: Thursday, April 17, 2014 7:21 AM

To: users@kafka.apache.org

Subject: Re: Cluster design distribution and JBOD vs RAID

Hey Bob,

thanks for your detailed response. I have added comments inline.

*On Wed, Apr 16, 2014 at 7:41 PM, Bello, Bob wrote:*

Perhaps as you consider the size of your cluster, a few questions about the kind of messaging you are looking at? I can use an example of what we do in our production environment while not going into specifics. These are just observations from an OPS perspective. (sorry for the wall of text.)

\* Size of messages (<100 bytes, <1kB, <10kB, <100kB, <1MB, <10MB, etc).

(we run messages size between a few bytes to over 100KB with a few at over 1MB).

BERT> We have several uses cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at uses cases that have varying message sizes of 200, 300, 1000, 2200 bytes

- \* Volume of messages per second (we produce over 15k per second and can consume over 100K per second when we are processing though some lag)

BERT> The use case we are looking at currently has hourly peaks of about 450K messages per second. For sizing we want to make sure we can support 900K . Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

- \* # of Producer clients (a few, a lot) (we have over 300 app servers the produce messages to the Kafka cluster)

- \*\* Not only does this affect Kafka broker performance but it can use a lot of TCP connections specially if you run a large Kafka cluster

BERT> our producer count will be low ...maybe 8-16 hosts.

- \* # of Consumer clients (a few, a lot) (we have less than 50 app servers that consume at this time)

- \*\* This also affects the # of TCP connections to Kafka brokers. (We have over 2400+ TCP connections to our cluster)

BERT> This will be much higher but not sure yet. We are also looking at replacing some legacy technology with storm so this is a bit up in the air right now.

- \* Will you compress your message before sending them to Kafka? (we have a mix of snappy, gzip and non-compressed messages depending on the application). This can affect your disk usage

BERT> We will use whatever performs best ;) My gut is that we will be using snappy

- \* Planned retention period. Longer retention period = more storage required. (we have varied retention periods per topic, between 10 days and 30 days).

- \* The number of topics per cluster. I believe Kafka scales well with the number of topics, however you have to worry about a few things:

- \*\* More topics, means slower migration/failover when Kafka brokers are shutdown or fail. This has caused us time out issues. Planned shutdown of a Kafka broker can take over 30 seconds to over 3 minutes. (We have over >10 and <50 topics. We are growing topics rapidly.)

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover adn compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

- \* The number of partitions per topic. More partitions per topic = more open file handles, (2 per log file, one for data and one more the index). We run average of 130 partitions. You have to consider your cardinality for your messages if order is important. Can you use a key that allows a good distribution across partitions while maintaining order? If all your message end up in just a few partitions within the topic then it's harder scale the consumption. This all depends on your use case.

BERT> We are lucky that order is not critical for our large feeds. It might sound like good rationale to scale the # of partitions for a

topic to a huge number (for just in case). I think it all depends.

\* How many consumer threads can consume a single topic? You can't go wider than the # of partitions however Kafka clients easily work with a large # of partitions with a few consumer threads.

\* Producer vs. Consumer size. Is your messaging flow Producer or Consumer heavy. Kafka is awesome and sending data to consumers that use "recent" data. Since Kafka uses memory mapped files, any data from Kafka that is in RAM will be very fast. (Our servers have 256GB of ram on them).

BERT> Our default config config has a 256GB of memory also. One thing I do want to test is impact on cluster of reading data not in memory. Have you done any testing like this?

\* Size of your cluster vs. the # of replicas. Larger # of Kafka brokers means more chance of failure within the cluster. Same kind of reason why you generally won't see a large RAID5 array. You get one failure before you lose data. If you decide to run a large cluster and # of replicas will be important. How much risk are you willing to take? (We run a 6 node cluster with a replica factor of 3. We can lose a total of two nodes before losing data).

BERT> Thanks for the datapoint. We were also planning to go with replication factor of 3

\* Are you running on native iron or virtualized? VM is generally lower performance but can generally spin up new instances faster upon failure. We run on native iron so we get excellent performance at the cost of longer lead times to provision new Kafka brokers.

BERT> We are big fans of vms...however kafka will be on physical

\* Networking. Are you are running 100mbit, 1gig or 10gib? You can only produce and consume so much data. Larger clusters let you run a total aggregate bandwidth. Don't forget about replication! Topic/partition leaders must replicate to all replica Kafkabrokers (hub/spoke). How long can you wait for replication to occur after a planned or un-planned outage? (We run >1Gig).

BERT> 10gb....so cheap now. I did cost analysis and found that a single 10gb port costs about the same as 2 x 1gig. Five times the bandwidth and less latency makes it no brainer. If your kafka hosts have multiple nics make sure they are using the right port. This one bit me for a little. (hostname config in the broker config)

\* Monitoring. Large # of Kafka brokers means more to monitor. Do you have a centralized monitoring app? Kafka provides a lot (huge!) JMX information. Making sense of it all can take some time.

BERT> We have not determined what to use just yet for monitoring. What are you guys using?

\* Disk I/O. JBOD vs. RAID. How much are you willing to tolerate failures? Do you have provisioned IO? (We run native iron and local disk in a RAID configuration. It was easier for us to manage a single mount point than a bunch in a JBOD configuration. We rely of local RAID and Kafka replication to keep enough copies of our data. We have a large amount of disk capacity. We can tolerate large re-replication events due to broker failure without affecting producer or consumer performance.)

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID

rebuild and if so did it negatively impact the cluster. The standard server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

\* Disk capacity / Kafka Broker capacity. Depending on your volume, message size and retention period, how much disk space will you need? (Using our "crystal ball tech(tm)" we decided over 20TB per Kafka broker would meet our needs. We will probably add Kafka brokers over adding disk as we outgrow this.)

BERT> I need a crystal ball ;)

\* Separate clusters to keep information separated? Do you have a use case for keeping customer data separate? Compliance use cases such as PCI or SOX? This may be a good reason to keep separate Kafka clusters. I assume that you already will keep separate clusters for DEV/QA/PROD.

BERT> yes DEV/QA/PROD completely separate

\* Zookeeper performance - 3 node, 5 node or 7 node. Less nodes, better performance. More nodes, better failure tolerance. We run 5 nodes with the transaction logs on SSD. Our ZK update performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

# of partitions per Topic debate:

Personally, I'm a proponent of larger # of partitions per topic without going way large. You can add Kafka Brokers to increase capacity and get more performance. However though it's possible to add partitions after a topic is created, it can cause issues with your key hashing depending on your message architecture.

\* Increasing # of brokers = easy

\* Increasing the # of partitions in a topic with data in it = hard

For us, we will be adding more topics and as we add additional messaging functionality.

Example:

130 partitions per topic / 6 brokers = 5 leader partitions per broker per topic. If you replicate 3 the you will end up with 3x active partitions per broker.

1024 partitions per topic / 24 brokers =~ 43 leader partitions per broker per topic.

BERT> Thanks for the example. Good to see others are using larger partition counts.

Final thoughts:

There's no magical formula for this as already stated in the wiki. It is a lot of trial and error. I will say that we went from a few 100 messages per second volume to over 40k per second by adding one application and our

Kafka cluster didn't even blink.

Kafka is awesome.

Btw, we're running 0.8.0.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert  
Corderman

Sent: Wednesday, April 16, 2014 11:58 AM

To: users@kafka.apache.org

Subject: Cluster design distribution and JBOD vs RAID

I am wondering what others are doing in terms of cluster separation. (if at all) For example let's say I need 24 nodes to support a given workload.

What are the tradeoffs between a single 24 node cluster vs 2 x 12 node clusters for example. The application I support can support separation of data fairly easily as the data is all processed in the same way but can be sharded isolated based on customers. I understand the standard tradeoffs, for example putting all your eggs in one basket but curious as if there are any details specific to Kafka in terms of cluster scale out.

Somewhat related is the use of RAID vs JBOD, I have reviewed the documents on the Kafka site and understand the tradeoff between space as well as sequential IO vs random and the fact a RAID rebuild might kill the system.

I am specifically asking the question as it relates to larger cluster and the impact on the number of partitions a topic might need.

Take an example of a 24 node cluster with 12 drives each the cluster would have 288 drives. To ensure a topic is distributed across all drives a topic would require 288 partitions. I am planning to test some of this but wanted to know if there was a rule of thumb. The following link

<https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-Howdolchoosethenumberofpartitionsfortopic>  
(<https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-Howdolchoosethenumberofpartitionsfortopic>)

?


Talks about supporting up to 10K partitions but its not clear if this is for a cluster as a whole vs topic based

Those of you running larger clusters what are you doing?

Bert

reply (mailto:users%40kafka.apache.org?In-Reply-  
To=%3C03EAA0700DF21749A1BCA520739FE43CC5873584%40MER2-  
EXCH07A3.echostar.com%3E&Subject=RE%3A%20Cluster%20design%20distribution%20and%20JBOD%20vs%  
| permalink (/p/kafka/users/144jb9qyss/cluster-design-distribution-and-jbod-vs-raid)

**Maxime Brugidou (/user/Maxime-Brugidou/PocJziEJyxSU94t**

 (/user/Maxime-  
Brugidou/PocJziEJyxSU94HmeeykPa)

Are you sure about that? Our latest tests show that loosing the drive in a  
jbod setup makes the broker fail (unfortunately).

*On Apr 18, 2014 9:01 PM, "Bello, Bob" wrote:*

Yes you would lose the topic/partitions on the drive. I'm not quite sure  
if Kafka can determine what topics/partitions are missing or not. I suggest  
you try testing it.

- Bob

-----Original Message-----

From: Andrew Otto

Sent: Friday, April 18, 2014 8:36 AM

To: users@kafka.apache.org

Subject: Re: Cluster design distribution and JBOD vs RAID

BOB> We are using RAID10. It was a requirement from our Unix guys. The  
rationale for this was we didn't want to lose just a disk and to have to  
rebuild/re-replicate 20TB of data. We haven't experienced any drive  
failures that I am aware of. We have had complete server failures, but the  
data was still good. I believe we have 10-4TB drives in a RAID10  
configuration. I/O performance is very good.

Just curious, would losing one disk in a JBOD setup really mean you'd have  
to  
re-replicate 20TB of data? If a single drive dies, wouldn't you only lose  
the partitions that happen to be on that drive?

*On Apr 17, 2014, at 8:00 PM, Bello, Bob wrote:*

Some feedback from your feedback.

BERT> We have several uses cases we are looking at kafka for. Today we are  
just using the file system to buffer data between our systems. We are  
looking at uses cases that have varying message sizes of 200, 300, 1000,  
2200 bytes

BOB> Since you are using small message size, watch out or large index

files. You can stuff a lot of messages in to the default log file size of 512MB. We use 1GB log files before rolling them.

BERT> The use case we are looking at currently has hourly peaks of about 450K messages per second. For sizing we want to make sure we can support 900K . Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

BOB> I believe LinkedIn has reported getting a throughput of 900k messages though a 6 node cluster. If you can achieve a flush rate of 100MB/s (which is easy for a good RAID setup) having a 12 node cluster should be doable. Remember when your topic/partition leadership is balance across the cluster (preferred replica election) you get to take advantage of all the brokers. Don't forget to architect for a failures. Can your cluster handle max throughput with two Kafka broker in an offline state?

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover and compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

BOB> Yes, that is what we have seen. The current controller architecture takes longer for Kafka nodes to fail over. It's not the # of topics, but the # of topic/partitions that have to move over. When a Kafka broker fails (planned or unplanned), the producers and the consumers have to pause for all the topic/partition pairs that were the leader for the off line Kafka broker and they have to move to another Kafka broker that is in ISR. By having lots of topics/partitions (we have many thousands), it can take a bit. Remember it's only a chunk, not all topics and partitions. This of course can change as the Kafka development team changes how this works. I highly recommend creating your topic and partition counts in DEV/QA and test this out. You will see a difference.

As for the amount of data in the topic/partition that is of no concern for failover. The Kafka broker will only failover those topics/partitions that are in ISR. Replication time once a Kafka broker is brought back online will depend on how far behind the Kafka broker is from the leader. This is delta in the offset. Planned shutdowns can be minutes, unplanned shutdowns/failures can take hours for our data to re-replicate.

BERT> Our default config has a 256GB of memory also. One thing I do want to test is impact on cluster of reading data not in memory. Have you done any testing like this?

BOB> Yes, it's about putting enough data to flush outside the OS file cache. But 512GB of data in your topics to make sure the data is not in the cache. Also, you can reset and/or use new consumer groups and make sure you read from the lowest-offset. Watch your iostats to see if you get lots of reads. On a normal Kafka cluster that is reading cached memory (for consumption), you will not see read IO. Assuming you don't have other processes on the system reading data (such as log aggregation). We see 30MB writes/flushes ever-other-second with 1-2% IO utilization.

BERT> We have not determined what to use just yet for monitoring. What

are you guys using?

BOB> We are using a commercial APM solution. It's an java agent the plugs into the JVM on boot time. This reads the JMX information as well as file I/O rates, NIO rates and GC. It sends to a centralized monitoring console. Google "Java APM" for some ideas.

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

BOB> We are using RAID10. It was a requirement from our Unix guys. The rationale for this was we didn't want to lose just a disk and to have to rebuild/re-replicate 20TB of data. We haven't experienced any drive failures that I am aware of. We have had complete server failures, but the data was still good. I believe we have 10-4TB drives in a RAID10 configuration. I/O performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

BOB> We started with SSD. Kafka brokers itself doesn't write a lot of data frequently (to zookeeper). It's really about how your consumers flush their offsets. This is assuming you will be using the high-level consumer client. If you are going to flush the offsets to zookeeper on every message consumed (to get best effort nearly-exactly-once processing). You will be writing a lot of data to zookeeper. On our 5 node zookeeper cluster, we are doing 300+ writes per second, and can spike up to many 1000's. Typically it's 1-2MBs data rate. The SSDs are under 2% I/O utilization. 200MB of ZK data, and we clean up the files once per hour. We run some consumers in batch and flush on time delay. Other consumers are flush per message processed. It's the flush per message that causes the high-volume. Push back on DEVs and software architecture if they want to flush per message. Do it where it's only absolutely necessary. :)

The high level Kafka consumer is good at "at least once" processing. Exactly once is a harder nut to crack. Exactly once processing may require some custom code around the low-level Kafka consumer client.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert Corderman

Sent: Thursday, April 17, 2014 7:21 AM

To: users@kafka.apache.org

Subject: Re: Cluster design distribution and JBOD vs RAID



Hey Bob,

thanks for your detailed response. I have added comments inline.

*On Wed, Apr 16, 2014 at 7:41 PM, Bello, Bob wrote:*

Perhaps as you consider the size of your cluster, a few questions about the kind of messaging you are looking at? I can use an example of what

we

do in our production environment while not going into specifics. These are

just observations from an OPS perspective. (sorry for the wall of text.)

\* Size of messages (<100 bytes, <1kB, <10kB, <100kB, <1MB, <10MB, etc).

(we run messages size between a few bytes to over 100KB with a few at over

1MB).

BERT> We have several use cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at use cases that have varying message sizes of 200, 300, 1000, 2200 bytes

\* Volume of messages per second (we produce over 15k per second and can consume over 100K per second when we are processing though some lag)

BERT> The use case we are looking at currently has hourly peaks of about 450K messages per second. For sizing we want to make sure we can support 900K. Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

\* # of Producer clients (a few, a lot) (we have over 300 app servers the produce messages to the Kafka cluster)

\*\* Not only does this affect Kafka broker performance but it can use a lot

of TCP connections specially if you run a large Kafka cluster

BERT> our producer count will be low ...maybe 8-16 hosts.

\* # of Consumer clients (a few, a lot) (we have less than 50 app servers that consume at this time)

\*\* This also affects the # of TCP connections to Kafka brokers. (We have over 2400+ TCP connections to our cluster)

BERT> This will be much higher but not sure yet. We are also looking at replacing some legacy technology with storm so this is a bit up in the air right now.

\* Will you compress your message before sending them to Kafka? (we have

a

mix of snappy, gzip and non-compressed messages depending on the application). This can affect your disk usage

BERT> We will use whatever performs best ;) My gut is that we will be using snappy

\* Planned retention period. Longer retention period = more storage required. (we have varied retention periods per topic, between 10 days

and

30 days).

\* The number of topics per cluster. I believe Kafka scales well with the number of topics, however you have to worry about a few things:

\*\* More topics, means slower migration/failover when Kafka brokers are shutdown or fail. This has caused us time out issues. Planned shutdown of a Kafka broker can take over 30 seconds to over 3 minutes. (We have over 10 and <50 topics. We are growing topics rapidly.)

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover and compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

\* The number of partitions per topic. More partitions per topic = more open file handles, (2 per log file, one for data and one more the index).

We run average of 130 partitions. You have to consider your cardinality for your messages if order is important. Can you use a key that allows a good distribution across partitions while maintaining order? If all your message end up in just a few partitions within the topic then it's harder scale the consumption. This all depends on your use case.

BERT> We are lucky that order is not critical for our large feeds. It might sound like good rationale to scale the # of partitions for a topic to a huge number (for just in case). I think it all depends.

\* How many consumer threads can consume a single topic? You can't go wider than the # of partitions however Kafka clients easily work with a large # of partitions with a few consumer threads.

\* Producer vs. Consumer size. Is your messaging flow Producer or Consumer heavy. Kafka is awesome and sending data to consumers that use "recent" data. Since Kafka uses memory mapped files, any data from Kafka that is in RAM will be very fast. (Our servers have 256GB of ram on them).

BERT> Our default config has a 256GB of memory also. One thing I do want to test is impact on cluster of reading data not in memory. Have you done any testing like this?

\* Size of your cluster vs. the # of replicas. Larger # of Kafka brokers means more chance of failure within the cluster. Same kind of reason why you generally won't see a large RAID5 array. You get one failure before you lose data. If you decide to run a large cluster and # of replicas will be important. How much risk are you willing to take? (We run a 6 node cluster with a replica factor of 3. We can lose a total of two nodes before losing data).

BERT> Thanks for the datapoint. We were also planning to go with replication factor of 3

\* Are you running on native iron or virtualized? VM is generally lower performance but can generally spin up new instances faster upon failure. We

run on native iron so we get excellent performance at the cost of longer lead times to provision new Kafka brokers.

BERT> We are big fans of vms...however kafka will be on physical \* Networking. Are you are running 100mbit, 1gig or 10gib? You can only produce and consume so much data. Larger clusters let you run a total aggregate bandwidth. Don't forget about replication! Topic/partition leaders must replicate to all replica Kafkabrokers (hub/spoke). How long can you wait for replication to occur after a planned or un-planned outage?

(We run >1Gig).

BERT> 10gb....so cheap now. I did cost analysis and found that a single 10gb port costs about the same as 2 x 1gig. Five times the bandwidth and less latency makes it no brainer. If your kafka hosts have multiple nics make sure they are using the right port. This one bit me for a little. (hostname config in the broker config)

\* Monitoring. Large # of Kafka brokers means more to monitor. Do you have

a centralized monitoring app? Kafka provides a lot (huge!) JMX information.

Making sense of it all can take some time.

BERT> We have not determined what to use just yet for monitoring. What are you guys using?

\* Disk I/O. JBOD vs. RAID. How much are you willing to tolerate failures?

Do you have provisioned IO? (We run native iron and local disk in a RAID configuration. It was easier for us to manage a single mount point than a

bunch in a JBOD configuration. We rely of local RAID and Kafka replication

to keep enough copies of our data. We have a large amount of disk capacity.

We can tolerate large re-replication events due to broker failure without

affecting producer or consumer performance.)

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

\* Disk capacity / Kafka Broker capacity. Depending on your volume, message

size and retention period, how much disk space will you need? (Using our "crystal ball tech(tm)" we decided over 20TB per Kafka broker would meet our needs. We will probably add Kafka brokers over adding disk as we

outgrow this.)

BERT> I need a crystal ball ;)

\* Separate clusters to keep information separated? Do you have a use case

for keeping customer data separate? Compliance use cases such as PCI or SOX? This may be a good reason to keep separate Kafka clusters. I assume that you already will keep separate clusters for DEV/QA/PROD.

BERT> yes DEV/QA/PROD completely separate

\* Zookeeper performance - 3 node, 5 node or 7 node. Less nodes, better performance. More nodes, better failure tolerance. We run 5 nodes with the

transaction logs on SSD. Our ZK update performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

# of partitions per Topic debate:

Personally, I'm a proponent of larger # of partitions per topic without going way large. You can add Kafka Brokers to increase capacity and get more performance. However though it's possible to add partitions after a topic is created, it can cause issues with your key hashing depending on your message architecture.

\* Increasing # of brokers = easy

\* Increasing the # of partitions in a topic with data in it = hard

For us, we will be adding more topics and as we add additional messaging functionality.

Example:

130 partitions per topic / 6 brokers = 5 leader partitions per broker  
per  
topic. If you replicate 3 the you will end up with 3x active partitions  
per  
broker.

1024 partitions per topic / 24 brokers ==~ 43 leader partitions per  
broker  
per topic.

BERT> Thanks for the example. Good to see others are using larger partition counts.

Final thoughts:

There's no magical formula for this as already stated in the wiki. It is a

lot of trial and error. I will say that we went from a few 100 messages  
per

second volume to over 40k per second by adding one application and our Kafka cluster didn't even blink.

Kafka is awesome.

Btw, we're running 0.8.0.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert

Corderman

Sent: Wednesday, April 16, 2014 11:58 AM

To: users@kafka.apache.org

Subject: Cluster design distribution and JBOD vs RAID

I am wondering what others are doing in terms of cluster separation.

(if at

all) For example let's say I need 24 nodes to support a given workload.

What are the tradeoffs between a single 24 node cluster vs 2 x 12 node

clusters for example. The application I support can support separation

of

data fairly easily as the data is all processed in the same way but can

be

sharded isolated based on customers. I understand the standard

tradeoffs,

for example putting all your eggs in one basket but curious as if there

are

any details specific to Kafka in terms of cluster scale out.

Somewhat related is the use of RAID vs JBOD, I have reviewed the documents

on the Kafka site and understand the tradeoff between space as well as

sequential IO vs random and the fact a RAID rebuild might kill the

system.

I am specifically asking the question as it relates to larger cluster

and

the impact on the number of partitions a topic might need.

Take an example of a 24 node cluster with 12 drives each the cluster would

have 288 drives. To ensure a topic is distributed across all drives a

topic would require 288 partitions. I am planning to test some of this

but

wanted to know if there was a rule of thumb. The following link

[https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-](https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdIchoosethenumberofpartitionsforatopic)

[HowdIchoosethenumberofpartitionsforatopic](https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdIchoosethenumberofpartitionsforatopic)

([https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-](https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdIchoosethenumberofpartitionsforatopic)

[HowdIchoosethenumberofpartitionsforatopic](https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdIchoosethenumberofpartitionsforatopic))

?

Talks about supporting up to 10K partitions but its not clear if this is


for a cluster as a whole vs topic based

Those of you running larger clusters what are you doing?

Bert

reply (mailto:users%40kafka.apache.org?In-Reply-To=%3CCAHRcGahb3J7-E4m%2BsPFNwV%2BWxGqwGvKfa8DugC1OzMqjV%3DQrUQ%40mail.gmail.com%3E&Subject=RE%3A%20permalink (/p/kafka/users/144j1dsjw/cluster-design-distribution-and-jbod-vs-raid)

**Jay Kreps (/user/Jay-Kreps/1mSSJpzibcbeZQNGAa2WPO)**

 (/user/Jay-Kreps/1mSSJpzibcbeZQNGAa2WPO)

I think we are saying the same thing. If any drive fails the broker is down. But when the drive is repaired only the data on the destroyed drive will need to be restored from replicas.

~Jay

On Fri, Apr 18, 2014 at 3:24 PM, Maxime Brugidou wrote:

Are you sure about that? Our latest tests show that loosing the drive in a JBOD setup makes the broker fail (unfortunately).

*On Apr 18, 2014 9:01 PM, "Bello, Bob" wrote:*

Yes you would lose the topic/partitions on the drive. I'm not quite sure if Kafka can determine what topics/partitions are missing or not. I suggest you try testing it.

- Bob

-----Original Message-----

From: Andrew Otto

Sent: Friday, April 18, 2014 8:36 AM

To: users@kafka.apache.org

Subject: Re: Cluster design distribution and JBOD vs RAID

BOB> We are using RAID10. It was a requirement from our Unix guys. The rationale for this was we didn't want to lose just a disk and to have to rebuild/re-replicate 20TB of data. We haven't experienced any drive failures that I am aware of. We have had complete server failures, but the data was still good. I believe we have 10-4TB drives in a RAID10 configuration. I/O performance is very good.

Just curious, would losing one disk in a JBOD setup really mean you'd have

to  
re-replicate 20TB of data? If a single drive dies, wouldn't you only lose the partitions that happen to be on that drive?

*On Apr 17, 2014, at 8:00 PM, Bello, Bob wrote:*

Some feedback from your feedback.

BERT> We have several uses cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at uses cases that have varying message sizes of 200, 300, 1000,

2200 bytes

BOB> Since you are using small message size, watch out or large index files. You can stuff a lot of messages in to the default log file size of 512MB. We use 1GB log files before rolling them.

BERT> The use case we are looking at currently has hourly peaks of about

450K messages per second. For sizing we want to make sure we can support

900K . Our larger feed in terms of size peaks at 450MBsec so we want to

make sure the cluster we build can support 900MBsec

BOB> I believe LinkedIn has reported getting a throughput of 900k messages though a 6 node cluster. If you can achieve a flush rate of 100MB/s (which is easy for a good RAID setup) having a 12 node cluster should be doable. Remember when your topic/partition leadership is balance across the cluster (preferred replica election) you get to take advantage of all the brokers. Don't forget to architect for a failures. Can your cluster handle max throughput with two Kafka broker in an offline state?

BERT> Are you implying that the number of topics has direct correlation

to

the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover and compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

BOB> Yes, that is what we have seen. The current controller architecture

takes longer for Kafka nodes to fail over. It's not the # of topics, but the # of topic/partitions that have to move over. When a Kafka broker fails (planned or unplanned), the producers and the consumers have to pause for all the topic/partition pairs that were the leader for the off line Kafka broker and they have to move to another Kafka broker that is in ISR. By having lots of topics/partitions (we have many thousands), it can take a bit. Remember it's only a chunk, not all topics and partitions. This of course can change as the Kafka development team changes how this works. I

highly recommend creating your topic and partition counts in DEV/QA and test this out. You will see a difference.

As for the amount of data in the topic/partition that is of no concern for failover. The Kafka broker will only failover those topics/partitions that are in ISR. Replication time once a Kafka broker is brought back online will depend on how far behind the Kafka broker is from the leader. This is delta in the offset. Planned shutdowns can be minutes, unplanned shutdowns/failures can take hours for our data to re-replicate.

BERT> Our default config config has a 256GB of memory also. One thing I

do want to test is impact on cluster of reading data not in memory. Have

you done any testing like this?

BOB> Yes, it's about putting enough data to flush outside the OS file cache. But 512GB of data in your topics to make sure the data is not in the cache. Also, you can reset and/or use new consumer groups and make sure you read from the lowest-offset. Watch your iostats to see if you get lots of reads. On a normal Kafka cluster that is reading cached memory (for consumption), you will not see read IO. Assuming you don't have other processes on the system reading data (such as log aggregation). We see 30MB writes/flushes ever-other-second with 1-2% IO utilization.

BERT> We have not determined what to use just yet for monitoring. What

are you guys using?

BOB> We are using a commercial APM solution. It's an java agent the plugs into the JVM on boot time. This reads the JMX information as well as file I/O rates, NIO rates and GC. It sends to a centralized monitoring console. Google "Java APM" for some ideas.

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not

worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs

to play catch up on a big backlog.

BOB> We are using RAID10. It was a requirement from our Unix guys. The rationale for this was we didn't want to lose just a disk and to have to rebuild/re-replicate 20TB of data. We haven't experienced any drive failures that I am aware of. We have had complete server failures, but the data was still good. I believe we have 10-4TB drives in a RAID10 configuration. I/O performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance



tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

BOB> We started with SSD. Kafka brokers itself doesn't write a lot of data frequently (to zookeeper). It's really about how your consumers flush their offsets. This is assuming you will be using the high-level consumer client. If you are going to flush the offsets to zookeeper on every message consumed (to get best effort nearly-exactly-once processing). You will be writing a lot of data to zookeeper. On our 5 node zookeeper cluster, we are doing 300+ writes per second, and can spike up to many 1000's. Typically it's 1-2MBs data rate. The SSDs are under 2% I/O utilization. 200MB of ZK data, and we clean up the files once per hour. We run some consumers in batch and flush on time delay. Other consumers are flush per message processed. It's the flush per message that causes the high-volume.

Push back on DEVs and software architecture if they want to flush per message. Do it where it's only absolutely necessary. :)

The high level Kafka consumer is good at "at least once" processing. Exactly once is a harder nut to crack. Exactly once processing may require some custom code around the low-level Kafka consumer client.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert Corderman  
Sent: Thursday, April 17, 2014 7:21 AM  
To: users@kafka.apache.org  
Subject: Re: Cluster design distribution and JBOD vs RAID

Hey Bob,

thanks for your detailed response. I have added comments inline.

*On Wed, Apr 16, 2014 at 7:41 PM, Bello, Bob wrote:*

Perhaps as you consider the size of your cluster, a few questions about

the kind of messaging you are looking at? I can use an example of what we

do in our production environment while not going into specifics. These are

just observations from an OPS perspective. (sorry for the wall of text.)

\* Size of messages (<100 bytes, <1kB, <10kB, <100kB, <1MB, <10MB, etc),

(we run messages size between a few bytes to over 100KB with a few at over

1MB).

BERT> We have several uses cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at uses cases that have varying message sizes of 200, 300, 1000,

2200 bytes

- \* Volume of messages per second (we produce over 15k per second and can consume over 100K per second when we are processing though some lag)

BERT> The use case we are looking at currently has hourly peaks of about 450K messages per second. For sizing we want to make sure we can support 900K . Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

- \* # of Producer clients (a few, a lot) (we have over 300 app servers produce messages to the Kafka cluster)

the

- \*\* Not only does this affect Kafka broker performance but it can use a lot of TCP connections specially if you run a large Kafka cluster

BERT> our producer count will be low ...maybe 8-16 hosts.

- \* # of Consumer clients (a few, a lot) (we have less than 50 app servers that consume at this time)

servers

- \*\* This also affects the # of TCP connections to Kafka brokers. (We have over 2400+ TCP connections to our cluster)

BERT> This will be much higher but not sure yet. We are also looking at replacing some legacy technology with storm so this is a bit up in the air right now.

- \* Will you compress your message before sending them to Kafka? (we have a mix of snappy, gzip and non-compressed messages depending on the application). This can affect your disk usage

BERT> We will use whatever performs best ;) My gut is that we will be using snappy

- \* Planned retention period. Longer retention period = more storage required. (we have varied retention periods per topic, between 10 days and 30 days).

- \* The number of topics per cluster. I believe Kafka scales well with the number of topics, however you have to worry about a few things:

- \*\* More topics, means slower migration/failover when Kafka brokers are shutdown or fail. This has caused us time out issues. Planned shutdown of a Kafka broker can take over 30 seconds to over 3 minutes. (We have over 10 and <50 topics. We are growing topics rapidly.)

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover adn compare to 500 topics with 1

million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

- \* The number of partitions per topic. More partitions per topic = more open file handles, (2 per log file, one for data and one more the index).

We run average of 130 partitions. You have to consider your cardinality

for

your messages if order is important. Can you use a key that allows a good

distribution across partitions while maintaining order? If all your message

end up in just a few partitions within the topic then it's harder

scale

the

consumption. This all depends on your use case.

BERT> We are lucky that order is not critical for our large feeds. It might sound like good rationale to scale the # of partitions for a topic to a huge number (for just in case). I think it all depends.

- \* How many consumer threads can consume a single topic? You can't go

wider

than the # of partitions however Kafka clients easily work with a large

#

of partitions with a few consumer threads.

- \* Producer vs. Consumer size. Is your messaging flow Producer or Consumer

heavy. Kafka is awesome and sending data to consumers that use "recent"

data. Since Kafka uses memory mapped files, any data from Kafka that is

in

RAM will be very fast. (Our servers have 256GB of ram on them).

BERT> Our default config config has a 256GB of memory also. One thing I

do want to test is impact on cluster of reading data not in memory.

Have

you done any testing like this?

- \* Size of your cluster vs. the # of replicas. Larger # of Kafka

brokers

means more chance of failure within the cluster. Same kind of reason why

you generally won't see a large RAID5 array. You get one failure before

you

lose data. If you decide to run a large cluster and # of replicas will be

important. How much risk are you willing to take? (We run a 6 node cluster

with a replica factor of 3. We can lose a total of two nodes before losing

data).

BERT> Thanks for the datapoint. We were also planning to go with replication factor of 3

- \* Are you running on native iron or virtualized? VM is generally lower performance but can generally spin up new instances faster upon failure. We

run on native iron so we get excellent performance at the cost of longer lead times to provision new Kafka brokers.

BERT> We are big fans of vms...however kafka will be on physical

- \* Networking. Are you are running 100mbit, 1gig or 10gib? You can only produce and consume so much data. Larger clusters let you run a total aggregate bandwidth. Don't forget about replication! Topic/partition leaders must replicate to all replica Kafkabrokers (hub/spoke). How long

can you wait for replication to occur after a planned or un-planned outage?

(We run >1Gig).

BERT> 10gb....so cheap now. I did cost analysis and found that a single 10gb port costs about the same as 2 x 1gig. Five times the bandwidth and less latency makes it no brainer. If your kafka hosts have multiple nics make sure they are using the right port. This one bit me for a little. (hostname config in the broker config)

- \* Monitoring. Large # of Kafka brokers means more to monitor. Do you have

a centralized monitoring app? Kafka provides a lot (huge!) JMX information.

Making sense of it all can take some time.

BERT> We have not determined what to use just yet for monitoring.

What are you guys using?

- \* Disk I/O. JBOD vs. RAID. How much are you willing to tolerate failures?

Do you have provisioned IO? (We run native iron and local disk in a RAID configuration. It was easier for us to manage a single mount point than a bunch in a JBOD configuration. We rely of local RAID and Kafka replication to keep enough copies of our data. We have a large amount of disk capacity.

We can tolerate large re-replication events due to broker failure without affecting producer or consumer performance.)

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard

server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

- \* Disk capacity / Kafka Broker capacity. Depending on your volume, message size and retention period, how much disk space will you need? (Using our "crystal ball tech(tm)" we decided over 20TB per Kafka broker would meet our needs. We will probably add Kafka brokers over adding disk as we outgrow this.)

BERT> I need a crystal ball ;)

- \* Separate clusters to keep information separated? Do you have a use case for keeping customer data separate? Compliance use cases such as PCI or SOX? This may be a good reason to keep separate Kafka clusters. I assume that you already will keep separate clusters for DEV/QA/PROD.

BERT> yes DEV/QA/PROD completely separate

- \* Zookeeper performance - 3 node, 5 node or 7 node. Less nodes, better performance. More nodes, better failure tolerance. We run 5 nodes with the transaction logs on SSD. Our ZK update performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

- # of partitions per Topic debate:  
Personally, I'm a proponent of larger # of partitions per topic without going way large. You can add Kafka Brokers to increase capacity and get more performance. However though it's possible to add partitions after a topic is created, it can cause issues with your key hashing depending on your message architecture.

- \* Increasing # of brokers = easy
- \* Increasing the # of partitions in a topic with data in it = hard

For us, we will be adding more topics and as we add additional messaging functionality.

Example:

130 partitions per topic / 6 brokers = 5 leader partitions per broker  
per  
topic. If you replicate 3 the you will end up with 3x active  
partitions  
per  
broker.

1024 partitions per topic / 24 brokers =~ 43 leader partitions per  
broker  
per topic.

BERT> Thanks for the example. Good to see others are using larger  
partition counts.

Final thoughts:

There's no magical formula for this as already stated in the wiki. It  
is a

lot of trial and error. I will say that we went from a few 100  
messages

per  
second volume to over 40k per second by adding one application and our  
Kafka cluster didn't even blink.

Kafka is awesome.

Btw, we're running 0.8.0.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of

Bert

Corderman

Sent: Wednesday, April 16, 2014 11:58 AM

To: users@kafka.apache.org

Subject: Cluster design distribution and JBOD vs RAID

I am wondering what others are doing in terms of cluster separation.  
(if at

all) For example let's say I need 24 nodes to support a given  
workload.

What are the tradeoffs between a single 24 node cluster vs 2 x 12 node  
clusters for example. The application I support can support  
separation

of  
data fairly easily as the data is all processed in the same way but  
can

be  
sharded isolated based on customers. I understand the standard  
tradeoffs,

for example putting all your eggs in one basket but curious as if there

are

any details specific to Kafka in terms of cluster scale out.

Somewhat related is the use of RAID vs JBOD, I have reviewed the documents

on the Kafka site and understand the tradeoff between space as well as sequential IO vs random and the fact a RAID rebuild might kill the system.

I am specifically asking the question as it relates to larger cluster and

the impact on the number of partitions a topic might need.

Take an example of a 24 node cluster with 12 drives each the cluster would

have 288 drives. To ensure a topic is distributed across all drives a topic would require 288 partitions. I am planning to test some of this

but

wanted to know if there was a rule of thumb. The following link

[https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-](https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdoIchoosethenumberofpartitionsfortopic)

[HowdoIchoosethenumberofpartitionsfortopic](https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdoIchoosethenumberofpartitionsfortopic)

([https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-](https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdoIchoosethenumberofpartitionsfortopic)

[HowdoIchoosethenumberofpartitionsfortopic](https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdoIchoosethenumberofpartitionsfortopic))

?

Talks about supporting up to 10K partitions but its not clear if this

is

for a cluster as a whole vs topic based

Those of you running larger clusters what are you doing?

Bert

reply (mailto:users%40kafka.apache.org?In-Reply-To=%3CCAOeJiJ8w2RivcANZvx-rRL8ioBviaezt43q0pxji\_fjstZpCA%40mail.gmail.com%3E&Subject=Re%3A%20Cluster%20design%20distribution%20and%20raid)  
| permalink (/p/kafka/users/144ktn4md/cluter-design-distribution-and-jbod-vs-raid)

Andrew Otto (/user/Andrew-Otto/ANomU8wPFR6gnymSg

(/user/Andrew-Otto/ANomU8wPFR6gnymSgRjQx9)

If any drive fails the broker is down.

Oh interesting.

Maybe it should only remove partitions on that drive from the ISR...feature request! :)

*On Apr 19, 2014, at 2:12 PM, Jay Kreps wrote:*

I think we are saying the same thing. If any drive fails the broker is down. But when the drive is repaired only the data on the destroyed drive will need to be restored from replicas.

-Jay

On Fri, Apr 18, 2014 at 3:24 PM, Maxime Brugidou wrote:

Are you sure about that? Our latest tests show that loosing the drive in a JBOD setup makes the broker fail (unfortunately).

*On Apr 18, 2014 9:01 PM, "Bello, Bob" wrote:*

Yes you would lose the topic/partitions on the drive. I'm not quite sure if Kafka can determine what topics/partitions are missing or not. I suggest you try testing it.

- Bob

-----Original Message-----

From: Andrew Otto

Sent: Friday, April 18, 2014 8:36 AM

To: users@kafka.apache.org

Subject: Re: Cluster design distribution and JBOD vs RAID

BOB> We are using RAID10. It was a requirement from our Unix guys. The rationale for this was we didn't want to lose just a disk and to have to rebuild/re-replicate 20TB of data. We haven't experienced any drive failures that I am aware of. We have had complete server failures, but the data was still good. I believe we have 10-4TB drives in a RAID10 configuration. I/O performance is very good.

Just curious, would losing one disk in a JBOD setup really mean you'd have to re-replicate 20TB of data? If a single drive dies, wouldn't you only lose the partitions that happen to be on that drive?

*On Apr 17, 2014, at 8:00 PM, Bello, Bob wrote:*



Some feedback from your feedback.

BERT> We have several uses cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at uses cases that have varying message sizes of 200, 300, 1000,

2200 bytes

BOB> Since you are using small message size, watch out or large index files. You can stuff a lot of messages in to the default log file size of 512MB. We use 1GB log files before rolling them.

BERT> The use case we are looking at currently has hourly peaks of about

450K messages per second. For sizing we want to make sure we can support

900K . Our larger feed in terms of size peaks at 450MBsec so we want to

make sure the cluster we build can support 900MBsec

BOB> I believe LinkedIn has reported getting a throughput of 900k messages though a 6 node cluster. If you can achieve a flush rate of 100MB/s (which is easy for a good RAID setup) having a 12 node cluster should be doable. Remember when your topic/partition leadership is balance across the cluster (preferred replica election) you get to take advantage of all the brokers. Don't forget to architect for a failures. Can your cluster handle max throughput with two Kafka broker in an offline state?

BERT> Are you implying that the number of topics has direct correlation to

the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover and compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

BOB> Yes, that is what we have seen. The current controller architecture

takes longer for Kafka nodes to fail over. It's not the # of topics, but the # of topic/partitions that have to move over. When a Kafka broker fails (planned or unplanned), the producers and the consumers have to pause for all the topic/partition pairs that were the leader for the off line Kafka broker and they have to move to another Kafka broker that is in ISR. By having lots of topics/partitions (we have many thousands), it can take a bit. Remember it's only a chunk, not all topics and partitions. This of course can change as the Kafka development team changes how this works. I highly recommend creating your topic and partition counts in DEV/QA and test this out. You will see a difference.

As for the amount of data in the topic/partition that is of no concern for failover. The Kafka broker will only failover those topics/partitions that are in ISR. Replication time once a Kafka broker is brought back online will depend on how far behind the Kafka broker is from the leader. This is delta in the offset. Planned shutdowns can be minutes, unplanned shutdowns/failures can take hours for our data to re-replicate.

BERT> Our default config has a 256GB of memory also. One thing I

do want to test is impact on cluster of reading data not in memory.

Have

you done any testing like this?

BOB> Yes, it's about putting enough data to flush outside the OS file cache. But 512GB of data in your topics to make sure the data is not in the cache. Also, you can reset and/or use new consumer groups and make sure you read from the lowest-offset. Watch your iostats to see if you get lots of reads. On a normal Kafka cluster that is reading cached memory (for consumption), you will not see read IO. Assuming you don't have other processes on the system reading data (such as log aggregation). We see 30MB writes/flushes every-other-second with 1-2% IO utilization.

BERT> We have not determined what to use just yet for monitoring.

What

are you guys using?

BOB> We are using a commercial APM solution. It's a Java agent that plugs into the JVM on boot time. This reads the JMX information as well as file I/O rates, NIO rates and GC. It sends to a centralized monitoring console. Google "Java APM" for some ideas.

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not

worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs

to play catch up on a big backlog.

BOB> We are using RAID10. It was a requirement from our Unix guys. The rationale for this was we didn't want to lose just a disk and to have to rebuild/re-replicate 20TB of data. We haven't experienced any drive failures that I am aware of. We have had complete server failures, but the data was still good. I believe we have 10-4TB drives in a RAID10 configuration. I/O performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at

zookeeper performance to see if its negatively impacting the performance

tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

BOB> We started with SSD. Kafka brokers itself doesn't write a lot of data frequently (to zookeeper). It's really about how your consumers flush their offsets. This is assuming you will be using the high-level consumer client. If you are going to flush the offsets to zookeeper on every message consumed (to get best effort nearly-exactly-once processing). You will

being writing a lot of data to zookeeper. On our 5 node zookeeper cluster, we are doing 300+ writes per second, and can spike up to many 1000's. Typically it's 1-2MBs data rate. The SSDs are under 2% I/O utilization. 200MB of ZK data, and we clean up the files once per hour. We run some consumers in batch and flush on time delay. Other consumers are flush per message processed. It's the flush per message that causes the high-volume.

Push back on DEVs and software architecture if they want to flush per message. Do it where it's only absolutely necessary. :)  
The high level Kafka consumer is good at "at least once" processing. Exactly once is a harder nut to crack. Exactly once processing may require some custom code around the low-level Kafka consumer client.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert Corderman  
Sent: Thursday, April 17, 2014 7:21 AM  
To: users@kafka.apache.org  
Subject: Re: Cluster design distribution and JBOD vs RAID

Hey Bob,

thanks for your detailed response. I have added comments inline.

On Wed, Apr 16, 2014 at 7:41 PM, Bello, Bob <bob.bello@dish.com> wrote:

Perhaps as you consider the size of your cluster, a few questions about

the kind of messaging you are looking at? I can use an example of what we do in our production environment while not going into specifics. These are just observations from an OPS perspective. (sorry for the wall of text.)

\* Size of messages (<100 bytes, <1kB, <10kB, <100kB, <1MB, <10MB, etc).

(we run messages size between a few bytes to over 100KB with a few at over 1MB).

BERT> We have several uses cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at uses cases that have varying message sizes of 200, 300, 1000,

2200 bytes

\* Volume of messages per second (we produce over 15k per second and can

consume over 100K per second when we are processing though some lag)

BERT> The use case we are looking at currently has hourly peaks of about

450K messages per second. For sizing we want to make sure we can

support  
900K . Our larger feed in terms of size peaks at 450MBsec so we want  
to  
make sure the cluster we build can support 900MBsec  
\* # of Producer clients (a few, a lot) (we have over 300 app servers  
the  
produce messages to the Kafka cluster)  
\*\* Not only does this affect Kafka broker performance but it can use a  
lot  
of TCP connections specially if you run a large Kafka cluster  
BERT> our producer count will be low ...maybe 8-16 hosts.  
\* # of Consumer clients (a few, a lot) (we have less than 50 app  
servers  
that consume at this time)  
\*\* This also affects the # of TCP connections to Kafka brokers. (We  
have  
over 2400+ TCP connections to our cluster)  
BERT> This will be much higher but not sure yet. We are also looking  
at  
replacing some legacy technology with storm so this is a bit up in the air  
right now.  
\* Will you compress your message before sending them to Kafka? (we  
have  
a  
mix of snappy, gzip and non-compressed messages depending on the  
application). This can affect your disk usage  
BERT> We will use whatever performs best ;) My gut is that we will be  
using snappy  
\* Planned retention period. Longer retention period = more storage  
required. (we have varied retention periods per topic, between 10 days  
and  
30 days).  
\* The number of topics per cluster. I believe Kafka scales well with  
the  
number of topics, however you have to worry about a few things:  
\*\* More topics, means slower migration/failover when Kafka brokers are  
shutdown or fail. This has caused us time out issues. Planned shutdown  
of a  
Kafka broker can take over 30 seconds to over 3 minutes. (We have over 10  
and <50 topics. We are growing topics rapidly.)  
BERT> Are you implying that the number of topics has direct  
correlation  
to  
the fail-over time? I think I might test this by creating one topic  
loading 500 million rows and test failover adn compare to 500 topics with 1  
million rows each. Not sure if data in the Q impacts the failover so  
figured I would test that also.  
\* The number of partitions per topic. More partitions per topic = more  
open file handles, (2 per log file, one for data and one more the  
index).  
We run average of 130 partitions. You have to consider your  
cardinality

for  
your messages if order is important. Can you use a key that allows a  
good  
distribution across partitions while maintaining order? If all your  
message  
end up in just a few partitions within the topic then it's harder  
scale  
the  
consumption. This all depends on your use case.  
BERT> We are lucky that order is not critical for our large feeds.  
It might sound like good rationale to scale the # of partitions for a  
topic to a huge number (for just in case). I think it all depends.  
  
\* How many consumer threads can consume a single topic? You can't go  
wider  
than the # of partitions however Kafka clients easily work with a  
large  
#  
of partitions with a few consumer threads.  
  
\* Producer vs. Consumer size. Is your messaging flow Producer or  
Consumer  
heavy. Kafka is awesome and sending data to consumers that use  
"recent"  
data. Since Kafka uses memory mapped files, any data from Kafka that  
is  
in  
RAM will be very fast. (Our servers have 256GB of ram on them).  
BERT> Our default config config has a 256GB of memory also. One  
thing I  
do want to test is impact on cluster of reading data not in memory.  
Have  
you done any testing like this?  
  
\* Size of your cluster vs. the # of replicas. Larger # of Kafka  
brokers  
means more chance of failure within the cluster. Same kind of reason  
why  
you generally won't see a large RAID5 array. You get one failure  
before  
you  
lose data. If you decide to run a large cluster and # of replicas will  
be  
important. How much risk are you willing to take? (We run a 6 node  
cluster  
with a replica factor of 3. We can lose a total of two nodes before  
losing  
data).  
BERT> Thanks for the datapoint. We were also planning to go with  
replication factor of 3  
  
\* Are you running on native iron or virtualized? VM is generally lower  
performance but can generally spin up new instances faster upon  
failure. We  
run on native iron so we get excellent performance at the cost of

longer  
lead times to provision new Kafka brokers.  
BERT> We are big fans of vms...however kafka will be on physical  
\* Networking. Are you are running 100mbit, 1gig or 10gib? You can only  
produce and consume so much data. Larger clusters let you run a total  
aggregate bandwidth. Don't forget about replication! Topic/partition  
leaders must replicate to all replica Kafkabrokers (hub/spoke). How  
long  
can you wait for replication to occur after a planned or un-planned  
outage?  
(We run >1Gig).  
BERT> 10gb....so cheap now. I did cost analysis and found that a  
single  
10gb port costs about the same as 2 x 1gig. Five times the bandwidth  
and  
less latency makes it no brainer. If your kafka hosts have multiple  
nics  
make sure they are using the right port. This one bit me for a little.  
(hostname config in the broker config)  
\* Monitoring. Large # of Kafka brokers means more to monitor. Do you  
have  
a centralized monitoring app? Kafka provides a lot (huge!) JMX  
information.  
Making sense of it all can take some time.  
BERT> We have not determined what to use just yet for monitoring.  
What  
are you guys using?  
\* Disk I/O. JBOD vs. RAID. How much are you willing to tolerate  
failures?  
Do you have provisioned IO? (We run native iron and local disk in a  
RAID  
configuration. It was easier for us to manage a single mount point  
than  
a  
bunch in a JBOD configuration. We rely of local RAID and Kafka  
replication  
to keep enough copies of our data. We have a large amount of disk  
capacity.  
We can tolerate large re-replication events due to broker failure  
without  
affecting producer or consumer performance.)  
BERT> Can you share more about your config? Are you using RAID10 or  
RAID5? What size and speed of drives? Have you needed to do a RAID  
rebuild and if so did it negatively impact the cluster. The standard  
server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or  
as RAID10. Parity based RAID with 4TB drives makes me nervous. I am  
not  
worried about performance when things are working as designed...we need to  
plan for edge cases when consumer is reading old data or the system  
needs  
to play catch up on a big backlog.  
\* Disk capacity / Kafka Broker capacity. Depending on your volume,

message  
size and retention period, how much disk space will you need? (Using  
our  
"crystal ball tech(tm)" we decided over 20TB per Kafka broker would  
meet  
our needs. We will probably add Kafka brokers over adding disk as we  
outgrow this.)  
BERT> I need a crystal ball ;)  
\* Separate clusters to keep information separated? Do you have a use  
case  
for keeping customer data separate? Compliance use cases such as PCI  
or  
SOX? This may be a good reason to keep separate Kafka clusters. I  
assume  
that you already will keep separate clusters for DEV/QA/PROD.  
BERT> yes DEV/QA/PROD completely separate  
\* Zookeeper performance - 3 node, 5 node or 7 node. Less nodes, better  
performance. More nodes, better failure tolerance. We run 5 nodes with  
the  
transaction logs on SSD. Our ZK update performance is very good.  
BERT> Need to spend some time on zookeeper. I have not looked at  
zookeeper performance to see if its negatively impacting the  
performance  
tests I am doing. We haven't spent any time looking at zookeeper. Did you  
find that the SSD helped improve kafka performance?  
# of partitions per Topic debate:  
Personally, I'm a proponent of larger # of partitions per topic  
without  
going way large. You can add Kafka Brokers to increase capacity and  
get  
more performance. However though it's possible to add partitions  
after a  
topic is created, it can cause issues with your key hashing depending  
on  
your message architecture.  
  
\* Increasing # of brokers = easy  
\* Increasing the # of partitions in a topic with data in it = hard  
  
For us, we will be adding more topics and as we add additional  
messaging  
functionality.  
  
Example:  
  
130 partitions per topic / 6 brokers = 5 leader partitions per broker  
per  
topic. If you replicate 3 the you will end up with 3x active  
partitions  
per  
broker.  
  
1024 partitions per topic / 24 brokers =~ 43 leader partitions per

broker

per topic.

BERT> Thanks for the example. Good to see others are using larger partition counts.

Final thoughts:

There's no magical formula for this as already stated in the wiki. It is a

lot of trial and error. I will say that we went from a few 100 messages

per

second volume to over 40k per second by adding one application and our Kafka cluster didn't even blink.

Kafka is awesome.

Btw, we're running 0.8.0.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of

Bert

Corderman

Sent: Wednesday, April 16, 2014 11:58 AM

To: users@kafka.apache.org

Subject: Cluster design distribution and JBOD vs RAID

I am wondering what others are doing in terms of cluster separation.

(if at

all) For example let's say I need 24 nodes to support a given workload.

What are the tradeoffs between a single 24 node cluster vs 2 x 12 node clusters for example. The application I support can support

separation

of

data fairly easily as the data is all processed in the same way but can

be

sharded isolated based on customers. I understand the standard tradeoffs,

for example putting all your eggs in one basket but curious as if there

are

any details specific to Kafka in terms of cluster scale out.

Somewhat related is the use of RAID vs JBOD, I have reviewed the documents



on the Kafka site and understand the tradeoff between space as well as sequential IO vs random and the fact a RAID rebuild might kill the system.

I am specifically asking the question as it relates to larger cluster and the impact on the number of partitions a topic might need.

Take an example of a 24 node cluster with 12 drives each the cluster would

have 288 drives. To ensure a topic is distributed across all drives a topic would require 288 partitions. I am planning to test some of this

but

wanted to know if there was a rule of thumb. The following link <https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-Howdolchoosethenumberofpartitionsfortopic> (<https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-Howdolchoosethenumberofpartitionsfortopic>)

?

Talks about supporting up to 10K partitions but its not clear if this is

for a cluster as a whole vs topic based

Those of you running larger clusters what are you doing?

Bert

reply (mailto:users%40kafka.apache.org?In-Reply-To=%3CD540328E-7E1B-4F35-9FF0-38EE12ECF2F6%40wikimedia.org%3E&Subject=Re%3A%20Cluster%20design%20distribution%20and%20raid) | permalink (/p/kafka/users/144nsqdb3f/cluster-design-distribution-and-jbod-vs-raid)

**Bert Corderman (/user/Bert-Corderman/bC2iM8XuTgD72smLK**



(/user/Bert-Corderman/bC2iM8XuTgD72smLKY7eZ5)

thanks again for your response and sorry for the delay, but I was out for a few days. I jsut wrapped up a bunch of performance testing, I will be sending link out to the user group here shortly.

Bert

*On Thu, Apr 17, 2014 at 8:00 PM, Bello, Bob wrote:*

Some feedback from your feedback.

BERT> We have several uses cases we are looking at kafka for. Today we are

just using the file system to buffer data between our systems. We are looking at uses cases that have varying message sizes of 200, 300, 1000, 2200 bytes

BOB> Since you are using small message size, watch out for large index files. You can stuff a lot of messages in to the default log file size of 512MB. We use 1GB log files before rolling them.

BERT> The use case we are looking at currently has hourly peaks of about 450K messages per second. For sizing we want to make sure we can support 900K . Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

BOB> I believe LinkedIn has reported getting a throughput of 900k messages though a 6 node cluster. If you can achieve a flush rate of 100MB/s (which is easy for a good RAID setup) having a 12 node cluster should be doable. Remember when your topic/partition leadership is balanced across the cluster (preferred replica election) you get to take advantage of all the brokers. Don't forget to architect for a failure. Can your cluster handle max throughput with two Kafka brokers in an offline state?

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover and compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

BOB> Yes, that is what we have seen. The current controller architecture takes longer for Kafka nodes to fail over. It's not the # of topics, but the # of topic/partitions that have to move over. When a Kafka broker fails (planned or unplanned), the producers and the consumers have to pause for all the topic/partition pairs that were the leader for the offline Kafka broker and they have to move to another Kafka broker that is in ISR. By having lots of topics/partitions (we have many thousands), it can take a bit. Remember it's only a chunk, not all topics and partitions. This of course can change as the Kafka development team changes how this works. I highly recommend creating your topic and partition counts in DEV/QA and test this out. You will see a difference.

As for the amount of data in the topic/partition that is of no concern for failover. The Kafka broker will only failover those topics/partitions that are in ISR. Replication time once a Kafka broker is brought back online will depend on how far behind the Kafka broker is from the leader. This is delta in the offset. Planned shutdowns can be minutes, unplanned shutdowns/failures can take hours for our data to re-replicate.

BERT> Our default config has a 256GB of memory also. One thing I do want to test is impact on cluster of reading data not in memory. Have you done any testing like this?

BOB> Yes, it's about putting enough data to flush outside the OS file

cache. But 512GB of data in your topics to make sure the data is not in the cache. Also, you can reset and/or use new consumer groups and make sure you read from the lowest-offset. Watch your iostats to see if you get lots of reads. On a normal Kafka cluster that is reading cached memory (for consumption), you will not see read IO. Assuming you don't have other processes on the system reading data (such as log aggregation). We see 30MB writes/flushes ever-other-second with 1-2% IO utilization.

BERT> We have not determined what to use just yet for monitoring. What are you guys using?

BOB> We are using a commercial APM solution. It's an java agent the plugs into the JVM on boot time. This reads the JMX information as well as file I/O rates, NIO rates and GC. It sends to a centralized monitoring console. Google "Java APM" for some ideas.

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

BOB> We are using RAID10. It was a requirement from our Unix guys. The rationale for this was we didn't want to lose just a disk and to have to rebuild/re-replicate 20TB of data. We haven't experienced any drive failures that I am aware of. We have had complete server failures, but the data was still good. I believe we have 10-4TB drives in a RAID10 configuration. I/O performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

BOB> We started with SSD. Kafka brokers itself doesn't write a lot of data frequently (to zookeeper). It's really about how your consumers flush their offsets. This is assuming you will be using the high-level consumer client. If you are going to flush the offsets to zookeeper on every message consumed (to get best effort nearly-exactly-once processing). You will be writing a lot of data to zookeeper. On our 5 node zookeeper cluster, we are doing 300+ writes per second, and can spike up to many 1000's. Typically it's 1-2MBs data rate. The SSDs are under 2% I/O utilization. 200MB of ZK data, and we clean up the files once per hour. We run some consumers in batch and flush on time delay. Other consumers are flush per message processed. It's the flush per message that causes the high-volume.

Push back on DEVs and software architecture if they want to flush per message. Do it where it's only absolutely necessary. :)

The high level Kafka consumer is good at "at least once" processing. Exactly once is a harder nut to crack. Exactly once processing may require some custom code around the low-level Kafka consumer client.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert Corderman

Sent: Thursday, April 17, 2014 7:21 AM

To: users@kafka.apache.org

Subject: Re: Cluster design distribution and JBOD vs RAID

Hey Bob,

thanks for your detailed response. I have added comments inline.

*On Wed, Apr 16, 2014 at 7:41 PM, Bello, Bob wrote:*

Perhaps as you consider the size of your cluster, a few questions about the kind of messaging you are looking at? I can use an example of what we do in our production environment while not going into specifics. These are just observations from an OPS perspective. (sorry for the wall of text.)

\* Size of messages (<100 bytes, <1kB, <10kB, <100kB, <1MB, <10MB, etc). (we run messages size between a few bytes to over 100KB with a few at over 1MB).

BERT> We have several uses cases we are looking at kafka for. Today we are just using the file system to buffer data between our systems. We are looking at uses cases that have varying message sizes of 200, 300, 1000, 2200 bytes

\* Volume of messages per second (we produce over 15k per second and can consume over 100K per second when we are processing though some lag)

BERT> The use case we are looking at currently has hourly peaks of about 450K messages per second. For sizing we want to make sure we can support 900K . Our larger feed in terms of size peaks at 450MBsec so we want to make sure the cluster we build can support 900MBsec

\* # of Producer clients (a few, a lot) (we have over 300 app servers the produce messages to the Kafka cluster)

\*\* Not only does this affect Kafka broker performance but it can use a lot of TCP connections specially if you run a large Kafka cluster

BERT> our producer count will be low ...maybe 8-16 hosts.

\* # of Consumer clients (a few, a lot) (we have less than 50 app servers that consume at this time)

\*\* This also affects the # of TCP connections to Kafka brokers. (We have over 2400+ TCP connections to our cluster)

BERT> This will be much higher but not sure yet. We are also looking at replacing some legacy technology with storm so this is a bit up in the air right now.

\* Will you compress your message before sending them to Kafka? (we have a mix of snappy, gzip and non-compressed messages depending on the application). This can affect your disk usage

BERT> We will use whatever performs best ;) My gut is that we will be using snappy

\* Planned retention period. Longer retention period = more storage required. (we have varied retention periods per topic, between 10 days and 30 days).

\* The number of topics per cluster. I believe Kafka scales well with the number of topics, however you have to worry about a few things:

\*\* More topics, means slower migration/failover when Kafka brokers are shutdown or fail. This has caused us time out issues. Planned shutdown of a Kafka broker can take over 30 seconds to over 3 minutes. (We have over 10 and <50 topics. We are growing topics rapidly.)

BERT> Are you implying that the number of topics has direct correlation to the fail-over time? I think I might test this by creating one topic loading 500 million rows and test failover and compare to 500 topics with 1 million rows each. Not sure if data in the Q impacts the failover so figured I would test that also.

\* The number of partitions per topic. More partitions per topic = more open file handles, (2 per log file, one for data and one more the index). We run average of 130 partitions. You have to consider your cardinality for your messages if order is important. Can you use a key that allows a good distribution across partitions while maintaining order? If all your message end up in just a few partitions within the topic then it's harder to scale the consumption. This all depends on your use case.

BERT> We are lucky that order is not critical for our large feeds. It might sound like good rationale to scale the # of partitions for a topic to a huge number (for just in case). I think it all depends.

\* How many consumer threads can consume a single topic? You can't go wider than the # of partitions however Kafka clients easily work with a large # of partitions with a few consumer threads.

\* Producer vs. Consumer size. Is your messaging flow Producer or Consumer heavy. Kafka is awesome and sending data to consumers that use "recent" data. Since Kafka uses memory mapped files, any data from Kafka that is in RAM will be very fast. (Our servers have 256GB of ram on them).

BERT> Our default config has a 256GB of memory also. One thing I do want to test is impact on cluster of reading data not in memory. Have you done any testing like this?

\* Size of your cluster vs. the # of replicas. Larger # of Kafka brokers means more chance of failure within the cluster. Same kind of reason why you generally won't see a large RAID5 array. You get one failure before you lose data. If you decide to run a large cluster and # of replicas will be important. How much risk are you willing to take? (We run a 6 node cluster with a replica factor of 3. We can lose a total of two nodes before losing data).

BERT> Thanks for the datapoint. We were also planning to go with replication factor of 3

\* Are you running on native iron or virtualized? VM is generally lower performance but can generally spin up new instances faster upon failure. We run on native iron so we get excellent performance at the cost of longer lead times to provision new Kafka brokers.

BERT> We are big fans of vms...however kafka will be on physical

\* Networking. Are you running 100mbit, 1gig or 10gib? You can only produce and consume so much data. Larger clusters let you run a total aggregate bandwidth. Don't forget about replication! Topic/partition leaders must replicate to all replica Kafkabrokers (hub/spoke). How long can you wait for replication to occur after a planned or un-planned outage? (We run >1Gig).

BERT> 10gb....so cheap now. I did cost analysis and found that a single 10gb port costs about the same as 2 x 1gig. Five times the bandwidth and less latency makes it no brainer. If your kafka hosts have multiple nics make sure they are using the right port. This one bit me for a little. (hostname config in the broker config)

\* Monitoring. Large # of Kafka brokers means more to monitor. Do you have a centralized monitoring app? Kafka provides a lot (huge!) JMX information.

Making sense of it all can take some time.

BERT> We have not determined what to use just yet for monitoring. What are you guys using?

\* Disk I/O. JBOD vs. RAID. How much are you willing to tolerate failures? Do you have provisioned IO? (We run native iron and local disk in a RAID configuration. It was easier for us to manage a single mount point than a bunch in a JBOD configuration. We rely on local RAID and Kafka replication

to keep enough copies of our data. We have a large amount of disk capacity. We can tolerate large re-replication events due to broker failure without affecting producer or consumer performance.)

BERT> Can you share more about your config? Are you using RAID10 or RAID5? What size and speed of drives? Have you needed to do a RAID rebuild and if so did it negatively impact the cluster. The standard server I was given has 12 x 4TB 7.2K drives. I will either run in JBOD or as RAID10. Parity based RAID with 4TB drives makes me nervous. I am not worried about performance when things are working as designed...we need to plan for edge cases when consumer is reading old data or the system needs to play catch up on a big backlog.

\* Disk capacity / Kafka Broker capacity. Depending on your volume, message size and retention period, how much disk space will you need? (Using our "crystal ball tech(tm)" we decided over 20TB per Kafka broker would meet our needs. We will probably add Kafka brokers over adding disk as we outgrow this.)

BERT> I need a crystal ball ;)

\* Separate clusters to keep information separated? Do you have a use case for keeping customer data separate? Compliance use cases such as PCI or SOX? This may be a good reason to keep separate Kafka clusters. I assume that you already will keep separate clusters for DEV/QA/PROD.

BERT> yes DEV/QA/PROD completely separate

\* Zookeeper performance - 3 node, 5 node or 7 node. Less nodes, better performance. More nodes, better failure tolerance. We run 5 nodes with the transaction logs on SSD. Our ZK update performance is very good.

BERT> Need to spend some time on zookeeper. I have not looked at zookeeper performance to see if its negatively impacting the performance tests I am doing. We haven't spent any time looking at zookeeper. Did you find that the SSD helped improve kafka performance?

# of partitions per Topic debate:

Personally, I'm a proponent of larger # of partitions per topic without going way large. You can add Kafka Brokers to increase capacity and get more performance. However though it's possible to add partitions after a topic is created, it can cause issues with your key hashing depending on your message architecture.

\* Increasing # of brokers = easy

\* Increasing the # of partitions in a topic with data in it = hard

For us, we will be adding more topics and as we add additional messaging functionality.

Example:

130 partitions per topic / 6 brokers = 5 leader partitions per broker per topic. If you replicate 3 the you will end up with 3x active partitions per broker.

1024 partitions per topic / 24 brokers =~ 43 leader partitions per broker per topic.

BERT> Thanks for the example. Good to see others are using larger partition counts.

Final thoughts:

There's no magical formula for this as already stated in the wiki. It is a lot of trial and error. I will say that we went from a few 100 messages per second volume to over 40k per second by adding one application and our Kafka cluster didn't even blink.

Kafka is awesome.

Btw, we're running 0.8.0.

- Bob

-----Original Message-----

From: bertcord@gmail.com On Behalf Of Bert Corderman

Sent: Wednesday, April 16, 2014 11:58 AM

To: users@kafka.apache.org

Subject: Cluster design distribution and JBOD vs RAID

I am wondering what others are doing in terms of cluster separation. (if at all) For example let's say I need 24 nodes to support a given workload. What are the tradeoffs between a single 24 node cluster vs 2 x 12 node

clusters for example. The application I support can support separation of data fairly easily as the data is all processed in the same way but can be sharded isolated based on customers. I understand the standard tradeoffs, for example putting all your eggs in one basket but curious as if there are any details specific to Kafka in terms of cluster scale out.

Somewhat related is the use of RAID vs JBOD, I have reviewed the documents on the Kafka site and understand the tradeoff between space as well as sequential IO vs random and the fact a RAID rebuild might kill the system. I am specifically asking the question as it relates to larger cluster and the impact on the number of partitions a topic might need.

Take an example of a 24 node cluster with 12 drives each the cluster would have 288 drives. To ensure a topic is distributed across all drives a topic would require 288 partitions. I am planning to test some of this but wanted to know if there was a rule of thumb. The following link

<https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdoIchoosethenumberofpartitionsfortopic>  
(<https://cwiki.apache.org/confluence/display/KAFKA/FAQ#FAQ-HowdoIchoosethenumberofpartitionsfortopic>)  
?

Talks about supporting up to 10K partitions but its not clear if this is for a cluster as a whole vs topic based

Those of you running larger clusters what are you doing?

Bert

reply (mailto:users%40kafka.apache.org?In-Reply-To=%3CCANL%3DHff5jd5NfVZ6eepgLssqF3PQ5Wti8q6DKgY6vSOMW-4piQ%40mail.gmail.com%3E&Subject=Re%3A%20Cluster%20design%20distribution%20and%20JBOD%20vs%20f | permalink (/p/kafka/users/144qqbjwb/cluster-design-distribution-and-jbod-vs-raid)

## Related Discussions

- Proper distribution of topics vs consumers (/t/kafka/users/134q1g2waa/proper-distribution-of-topics-vs-consumers)
- Log4J setting for kafka (/t/kafka/users/1371ck2aq3/log4j-setting-for-kafka)
- Differences in size of data replicated by mirror maker (/t/kafka/users/138pgtn5z1/differences-in-size-of-data-replicated-by-mirror-maker)
- Java vs. C++ in Client API (/t/kafka/users/143n8nvkdj/java-vs-c-in-client-api)
- Re: Kafka Sync vs. Async etc. (/t/kafka/users/1431p4g3dk/kafka-sync-vs-async-etc)
- LinkedIn Camus vs kafka-hadoop-loader vs hadoop-consumer (/t/kafka/users/142dxx02jp/linkedin-camus-vs-kafka-hadoop-loader-vs-hadoop-consumer)



- [Hardware Loadbalancer vs zookeeper for producers \(/t/kafka/users/139brrfz10/hardware-loadbalancer-vs-zookeeper-for-producers\)](/t/kafka/users/139brrfz10/hardware-loadbalancer-vs-zookeeper-for-producers)
- [Metrics: via Broker vs. Producer vs. Consumer \(/t/kafka/users/137r3rv0y8/metrics-via-broker-vs-producer-vs-consumer\)](/t/kafka/users/137r3rv0y8/metrics-via-broker-vs-producer-vs-consumer)
- [Zookeeper vs statics list of brokers \(/t/kafka/users/1378e60enb/zookeeper-vs-statics-list-of-brokers\)](/t/kafka/users/1378e60enb/zookeeper-vs-statics-list-of-brokers)
- [Kafka Monitoring, 0.7 vs. 0.8 JMX \(/t/kafka/users/13570qqrqq/kafka-monitoring-0-7-vs-0-8-jmx\)](/t/kafka/users/13570qqrqq/kafka-monitoring-0-7-vs-0-8-jmx)