Real-world OOM Errors in Distributed Data-parallel Applications

Lijie Xu

Institute of Software, Chinese Academy of Sciences

Abstract: This study aims to summarize root causes and fix methods of OOM errors in real-world MapReduce/Spark applications. These cases come from StackOverflow.com, Hadoop/Spark mailing list, developer's blogs, and two popular MapReduce books. The two MapReduce books are Data-Intensive Text Processing with MapReduce and MapReduce Design Patterns. The summarized causes and cause patterns are illustrated in the following table.

We totally found 276 cases. The causes of 123 cases have been identified by experts (Hadoop/Spark committers, experienced developers, or the book authors), users themselves or us in the reproduced errors. The causes of the left 153 cases are unknown from the users' error descriptions or the expert's answers.

TABLE I: DISTRIBUTION OF THE OOM ERRORS

Framework	Sources	Row code	Pig	Hive	Mahout	Cloud9	GraphX	MLlib	Total	Reproduced
Hadoop	StackOverflow.com	20	4	2	4	0	0	0	30	16
	Hadoop mailing list	5	5	1	0	1	0	0	12	6
	Developers' blogs	2	1	0	0	0	0	0	3	2
	MapReduce books	8	3	0	0	0	0	0	11	2
Spark	Spark mailing list	16	0	0	0	0	1	2	19	3
	StackOverflow.com	42	0	0	0	0	1	5	48	14
Total	All sources	93	13	3	4	1	2	7	123	43

TALBE II: OOM CAUSE PATTERNS

Category	Pattern	Pattern description	Hadoop	Spark	Total	Ratio
Large data stored in framework	Large buffered data	Large intermediate data stored in buffers	6	2	8	6%
	Large cached data	Large data cached in memory for reuse	0	7	7	6%
	subtotal		**6**	**9**	**15**	**12%**
Abnormal dataflow	Improper data partition	Some data partitions are extremely large	3	13	16	13%
	Hotspot key	Large (k, list(v))	15	8	23	18%
	Large single record	Large (k, v)	6	1	7	6%
	subtotal		**24**	**22**	**46**	**37%**
Memory-consuming user code	Large external data	User code loaded large external data	8	0	8	6%
	Large intermediate results	Large computing results are generated during processing a record	4(3)	2	6(3)	5%
	Large accumulated results	Large computing results are accumulated in user code	30[13]	10[1]	40[4]	33%
	Large data generated in driver	The driver generates large data in memory	0	9	9	7%
	Large results collected by driver	The dirver collects large results of the tasks	0	16	16	13%
	subtotal		**42**	**37**	**79**	**64%**

Notation: 4(3) means 3 out of the 4 errors also have the *large single record* cause pattern. 30[13] means 13 out of the 30 errors also have the *Hotspot key* cause pattern.

Contents

1. Large buffered data

1.1 Hadoop Errors (6)

1. Q: CDH 4.1: Error running child : java.lang.OutOfMemoryError: Java heap space

User: I've trying to overcome sudden problem. Before that problem I've used old VM. I've downloaded the new one VM and still can't make my job run. I get Java heap space error. I've already read this one post: out of Memory Error in Hadoop

Expert: lower the buffer size. a combination of 256 JVM to 128 sort could be your problem. **Try an io.sort.mb size of 64mb** – One could also try **setting the mapred.job.shuffle.input.buffer.percent to 20%. By default, this is set to 70%**, which could be a lot if you are working on a very large set of data.

Job type: User-defined (StackOverflow)
Causes: Large framework buffer (identified by expert, Reproduced)
Fix suggestions: lower buffer size (accepted)
Fix details: Set io.sort.mb from 128MB to 64MB (fixed)

2. Q: Out of memory error in Mapreduce shuffle phase

User: I am getting strange errors while running a **wordcount-like** mapreduce program. I have a hadoop cluster with 20 slaves, each having 4 GB RAM. I configured my map tasks to have a heap of 300MB and my reduce task slots get 1GB. I have 2 map slots and 1 reduce slot per node. Everything goes well until the first round of map tasks finishes. Then there progress remains at 100%. I suppose then the **copy phase** is taking place. Each map task generates something like: I think this explains why **the job no longer crashes if I lower the shuffle.input.buffer.percent**.
I think the clue is that the heapsize of my reduce task was required almost completely for the reduce phase. But the **shuffle phase is competing for the same heapspace**, the conflict which arose caused my jobs to crash.

Job type: User-defined (StackOverflow)
Causes: Large framework buffer (identified by user, Reproduced)
Fix suggestions: lower buffer size (user)
Fix details: Lower shuffle.input.buffer.percent from 0.7 to 0.2 (fixed)

3. Q: pig join gets OutOfMemoryError in reducer when mapred.job.shuffle.input.buffer.percent=0.70

User: We're doing a simple pig join between a small table and a big skewed table. We cannot use "using skewed" due to another bug (pig skewed join with a big table causes "Split metadata size exceeded 10000000") :(
If we use the default mapred.job.shuffle.input.buffer.percent=0.70 some of our reducers fail in the shuffle stage:
Expert: As you mentioned, when mapred.job.shuffle.input.buffer.percent=0.30, only 30% heap is used for storing shuffled data, heap is hard to be full.

Job type: Pig (StackOverflow)
Causes: Large framework buffer (identified by user, Reproduced)
Fix suggestions: lower buffer size (user), use "skewed"
Fix details: Change shuffle.input.buffer.percent from 0.7 to 0.3 (fixed), although in 2 hours.

4. A: out of Memory Error in Hadoop

User: I tried installing Hadoop following thishttp://hadoop.apache.org/common/docs/stable/single_node_setup.html document. When I tried executing this bin/hadoop jar hadoop-examples-*.jar grep input output 'dfs[a-z.]+'

Job type: User-defined (StackOverflow)
Causes: Large framework buffer (reproduced)
Fix suggestions: lower buffer size (us)
Fix details: Change io.sort.mb from 400MB to 200MB (fixed by us)

5. A: running an elementary mapreduce job with java on hadoop

User: The assignment is to run:
bin/hadoop jar hadoop-cookbook-chapter1.jar chapter1.WordCount input output
And this is the response that I get:
Expert: add heap size

Job type: User-defined (StackOverflow)
Causes: Large framework buffer (reproduced)
Fix suggestions: lower buffer size (us)
Fix details: Change io.sort.mb from 500MB to 200MB (fixed by us)

6. Out of heap space errors on TTs

User: I am running hive and I am trying to join two tables (2.2GB and 136MB) on a cluster of 9 nodes (replication = 3)
Expert: by default it will be 200mb. But your io.sort.mb(300) is more than that. So, configure more heap space for child tasks.

Job type: Apache Hive (Mailing list)
Causes: Large framework buffer (Expert)
Fix suggestions: lower buffer size (Expert)
Fix details: Lower io.sort.mb (from 300MB, suggestion)

1.2 Spark Errors (2)

1. A: org.apache.spark.shuffle.MetadataFetchFailedException: Missing an output location for shuffle 0

User:
We used `JavaPairRDD.repartitionAndSortWithinPartitions` on 100GB data and it kept failing similarly to your app. Then we looked at the Yarn logs on the specific nodes and found out that we have some kind of out-of-memory problem, so the Yarn interrupted the execution. Our solution was to change/add `spark.shuffle.memoryFraction 0` in `.../spark/conf/spark-defaults.conf`. That allowed us to handle a much larger (but unfortunately not infinite) amount of data this way.

Job type: User-defined (StackOverflow)
Causes: Large buffered data (User)
Fix suggestions: lower the buffer size (user)
Fix details: Change spark.shuffle.memoryFraction to 0 to force it to spill permanently, to force it to spill permanently, The price was 20% of time.

2. MLLib /ALS : java.lang.OutOfMemoryError: Java heap space

User: I am running into an out of memory error while running ALS using MLLIB on a reasonably small data set consisting of around 6 Million ratings.
at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:158)
at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:99)

Expert: I am not sure this can help you. I have 57 million rating,about 4million user and 4k items. I used 7-14 total-executor-cores,executal-memory 13g,cluster have 4 nodes,each have 4cores,max memory 16g.
I found set as follows may help avoid this problem:

```
    conf.set("spark.shuffle.memoryFraction","0.65") //default is 0.2
    conf.set("spark.storage.memoryFraction","0.3")//default is 0.6
```

I have to set rank value under 40, otherwise occure this problem.

Job type: User-defined (Mailing list)
Causes: Large buffered data (User)
Fix suggestions: decrease the rank value, lower the buffer size, lower the cache limit (Expert)
Fix details: decrease the rank value, I have to set rank value under 40

2. Large cached data

2.1 Spark Errors (7)

1. tiers of caching

User: i noticed that some algorithms such as graphx liberally cache RDDs for efficiency, which makes sense. however it can also leave a long trail of unused yet cached RDDs, that might push other RDDs out of memory.
Expert: I think tiers/priorities for caching are a very good idea and I'd be interested to see what others think. In addition to letting libraries cache RDDs liberally, it could also unify memory management across other parts of Spark. For example, small shuffles benefit from explicitly keeping the shuffle outputs in memory rather than writing it to disk, possibly due to filesystem overhead. To prevent in-memory shuffle outputs from competing with application RDDs, Spark could mark them as lower-priority and specify that they should be dropped to disk when memory runs low.

Job type: User-defined (Mailing list)
Causes: Large cached data (User)
Fix suggestions: no

2. Kyro serialization slow and runs OOM

User: when I load my dataset, transform it with some one to one transformations, and try to **cache the eventual RDD** - it runs really slow and then runs out of memory. When I remove Kyro serializer and default back to java serialization it works just fine and is able to load and cache the 700Gs of resultant data.

Job type: User-defined (Mailing list)
Causes: Large cached data (User)
Fix suggestions: no

3. Problems with broadcast large datastructure

User: Spark repeatedly fails broadcast a large object on a cluster of 25 machines for me. I have a node of 20 machines, and I just run the broadcast example, what I do is just change the data size in the example, to 400M, this is really a small data size. but I occurred the same problem with you .
Expert: 400MB isn't really that big. Broadcast is expected to work with several GB of data and in even larger clusters (100s of machines).
if you are using the default HttpBroadcast, then akka isn't used to move the broadcasted data. **But block manager can run out of memory if you repetitively broadcast large objects.** Another scenario is that the master isn't receiving any heartbeats from the blockmanager because the control messages are getting dropped due to bulk data movement. Can you provide a bit more details on your network setup?

Also, you can try doing a binary search over the size of broadcasted data to see at what size it breaks (i.e, try to broadcast 10, then 20, then 40 etc etc.)? Also, limit each run to a single iteration in the example (right now, it tries to broadcast 3 consecutive times).

If you are using a newer branch, you can also try the new TorrentBroadcast implementation.
your code is broadcasting 400MB 30 times, which are not being evicted from the cache fast enough, which, I think, is causing blockManagers to run out of memory.

Job type: User-defined (Mailing list)
Causes: Large cached data (Expert)
Fix suggestions: Try TorrentBroadcast

4. Spark streaming questions

User: Can someone explain the usage of cache w.r.t spark streaming? For example if we do stream.cache(), will the cache remain constatnt with all the partitions of rdd present across the nodes for that stream, OR will it be regularly updated as in while new batch is coming?
Expert: If you call DStream.persist (persist == cache = true), then all RDDs generated by the DStream will be persisted in the cache (in the BlockManager). As new RDDs are generated and persisted, old RDDs from the same DStream will fall out of memory. either by LRU or explicitly if spark.streaming.unpersist is set to true.
Well it is clear that the combineByKey is taking the most amount of time and 7 seconds. So you need to increase the number of reducers in the reduceByKeyAndWindow operation. That should distribute the computation more to use all the cores, and therefore speed up the processing of each batch.

Job type: Streaming (Mailing list)
Causes: Large cached data (Expert)
Fix suggestions: Change storage level, increase reduce number, add combineByKey()
Fix details: To fall of to disk you have to use MEMORY_AND_DISK_SER or MEMORY_AND_DISK_SER_2. Note that, SER = keep data serialized, good for GC behavior (see programming guide), and _2 = replicate twice.

5. OOM when calling cache on RDD with big data

User: I have a very simple job that simply caches the hadoopRDD by calling cache/persist on it. I tried MEMORY_ONLY, MEMORY_DISK and DISK_ONLY for caching strategy, I always get OOM on executors. And it works fine if I do not call cache or persist on the RDD:
Expert:

Job type: User-defined (Mailing list)
Causes: Large cached data (User)
Fix suggestions: no

6. [Graphx] some problem about using SVDPlusPlus

User: The implementation of SVDPlusPlus shows that it produces two new graph in each iteration which will also be cached to memory. However, **as the iteration goes on, more and more graph will be cached and out of memory happens.** So I think it maybe need to unpersist old graph which will not be used any more and add a few lines of code, the details are showed as follows:
Expert:

Job type: GraphX (Mailing list)
Causes: Large cached data (User)

Fix suggestions: unpersist old graph which will not be used any more

7. [0.9.0] MEMORY*AND*DISK_SER not falling back to disk

User: My understanding of the MEMORY_AND_DISK_SER persistence level was that if an RDD could fit into memory then it would be left there (same as MEMORY_ONLY), and only if it was too big for memory would it spill to disk. Here's how the docs describe it:
What I'm observing though is that really large RDDs are actually causing OOMs. I'm not sure if this is a regression in 0.9.0 or if it has been this way for some time.
I dropped down to 0.5 but still OOM'd, so sent it all the way to 0.1 and didn't get an OOM.
Expert: This probably means that there's not enough free memory for the "scratch" space used for computations, so we OOM before the Spark cache decides that it's full and starts to spill stuff. Try reducing spark.storage.memoryFraction (default is 0.66, try 0.5).

Job type: User-defined (Mailing list)
Causes: Large cached data (Expert)
Fix suggestions: lower the cache size
Fix details: dropped spark.storage.memoryFraction down to 0.5 but still OOM'd, so sent it all the way to 0.1 and didn't get an OOM. (fixed)

3. Improper data partition

3.1 Hadoop Errors (3)

1. Subject: Reducers fail with OutOfMemoryError while copying Map outputs

User: Hi , I am using M7,Reducers fail while copying Map outputs with following exception:
View Diagnostics:
 Error: java.lang.OutOfMemoryError: Java heap space at org.apache.hadoop.mapred.ReduceTask$ReduceCopier$MapOutputCopier.shuffleInMemory(ReduceTask.java:1774) at org.apache.hadoop.mapred.ReduceTask$ReduceCopier$MapOutputCopier.getMapOutputFromFile(ReduceTask.java:1487) at org.apache.hadoop.mapred.ReduceTask$ReduceCopier$MapOutputCopier.copyOutput(ReduceTask.java:1361)
Expert: What I see is that in the second map-reduce, you have 8 reducers, of which 7 completed successfully. 1 reducer failed, presumably this is the out-of-memory task.
Since this sounds like a problem that depends on the volume of input, I suspect that something about **your query has trouble with a very large number of items being sent to a single reducer. This can happen a number of different ways, but the problem of skew in data volumes to reducers is a very common one.** There are often clever tricks to avoid the OOM error that this can cause.

Job type: Pig (Mailing list)
Causes: Improper data partition (Expert, Reproduced)
Fix suggestions: no

2. Q: Reducer's Heap out of memory

User: So I have a few Pig scripts that keep dying in there reduce phase of the job with the errors that the Java heap keeps running out of space. To this date **my only solution has been to increase Reducer counts**, but that doesn't seem to be getting me anywhere reliable. Now part of this may be just the massive growth in data we are getting, but can't be sure.
Expert: Obviously you are running out of memory somewhere. **Increasing the number of reducers is actually quite reasonable.** Take a look at the stats on the JobTracker Web GUI and see how many bytes are going out of the mapper. Divide that by the number of reduce tasks, and that is a pretty rough estimate of what each reducer is getting. Unfortunately, this only works in the long run if your keys are evenly distributed.

Job type: Pig script (StackOverflow)
Causes: Improper data partition (user)
Fix suggestions: add reduce number and lower framework buffer (User)
Fix details: increase the reduce number (partially fixed)

3. Subject: OutOfMemoryError in ReduceTask shuffleInMemory

User/Expert: We were able to capture a heap dump of one reduce task. The heap contained 8 byte arrays **that were 127 MB each.** These byte arrays were all referenced by their own DataInputBuffer. Six of the buffers were referenced by the linked lists in ReduceTask$ReduceCopier.mapOutputsFilesInMemory. **These six byte arrays consume 127 MB * 6 = 762 MB of the heap**. Curiously, this 762 MB exceeds the 717 MB limit. The ShuffleRamManager.fullSize = 797966777 = 761MB, so something is a bit off in my original value of 717... But this is not the major source of trouble.

Job type: User-defined (Mailing list)
Causes: Improper data partition (us, Reproduced)
Fix suggestions: add reduce number (us)
Fix details: increase the reduce number (fixed by us)

3.2 Spark Errors (13)

1. Q: Spark runs out of memory when grouping by key

User:
I am attempting to perform a simple transformation of common crawl data using Spark host on an EC2 using this guide, my code looks like this:

So my basic question is, what is necessary to write a Spark task that can group by key with an almost unlimited amount of input without running out of memory?

Expert:
The most common cause of java.lang.OutOfMemoryError exceptions in shuffle tasks (such as groupByKey, reduceByKey, etc.) is low level of parallelism.

Job type: User-defined (StackOverflow)
Causes: Improper data partition (Expert)
Fix suggestions: add partition number
Fix details: Increase default value by setting spark.default.parallelism property (suggestion)

2. Q: Regarding spark input data partition and coalesce

User:
partition the input data(80 million records) into partitions using RDD.coalesce(numberOfPArtitions) before submitting it to mapper/reducer function. Without using coalesce() or repartition() on the input data spark executes really slow and fails with out of memory exception.

Expert:
Determining the number of partitions is a bit tricky. Spark by default will try and infer a sensible number of partitions.

Job type: User-defined (StackOverflow)
Causes: Improper data partition (User)
Fix suggestions: repartition() (User)

3. Q: Why does Spark RDD partition has 2GB limit for HDFS

User:
The Integer.MAX_SIZE is 2GB, it seems that some partition out of memory. So i repartiton my rdd partition to 1000, so that each partition could hold far less data as before. Finally, the problem is solved!!!
Expert:
It is a critical issue which prevents use of spark with very large datasets. Increasing the number of partitions can resolve it (like in OP's case), but is not always feasible, for instance when there is large chain of transformations part of which can increase data (flatMap etc) or in cases where data is skewed.

Job type: User-defined (StackOverflow)
Causes: Improper data partition (User)
Fix suggestions: add partition number (User) (not always feasible)

4. A: Spark Java Error: Size exceeds Integer.MAX_VALUE

User:
I am trying to use spark for some simple machine learning task. I used pyspark and spark 1.2.0 to do a simple logistic regression problem. I have 1.2 million records for training, and I hashed the features of the records. When I set the number of hashed features as 1024, the program works fine, but when I set the number of hashed features as 16384, the program fails several times with the following error:

Expert:
I'm no Python coder, but when you "hashed the features of the records" you might be taking a very sparse set of records for a sample and creating an non-sparse array. This will mean a lot of memory for 16384 features. Particularly, when you do `zip(line[1].indices, line[1].data)`. The only reason that doesn't get you out of memory right there is the shitload of it you seem to have configured (50G).
Reason:

Thanks. This problem is just fixed by **using more partitions when loading the data**. We are just testing on small data set and gain some idea, then we are going to apply to big data set with much powerful machine.

Job type: User-defined (StackOverflow)
Causes: Improper data partition (User)
Fix suggestions: add partition number (User fixed)

5. Q: Spark out of memory

User: The code I'm using: - reads TSV files, and extracts meaningful data to (String, String, String) triplets - afterwards some **filtering, mapping and grouping** is performed - finally, the data is reduced and some aggregates are calculated

I've been able to run this code with a single file (~200 MB of data), however I get a java.lang.OutOfMemoryError: GC overhead limit exceeded and/or a Java out of heap exception when adding more data (the application breaks with 6GB of data but I would like to use it with 150 GB of data).

I guess I would have to tune some parameters to make this work. I would appreciate any tips on how to approach this problem (how to debug for memory demands). I've tried increasing the 'spark.executor.memory' and using a smaller number of cores (the rational being that each core needs some heap space), but this didn't solve my problems.

typically, when **loading data from disk into a Spark RDD, the data consumes much more space in RAM than on disk**. This is paritally due to the overhead of making byte arrays into Java String objects.

I was thinking of something in the way of taking a chunk of data, processing it, storing partial results on disk (if needed), continuing with the next chunk until all are done, and finally merging partial results in the end.

Expert: If you repartition an RDD, it requires additional computation that has overhead above your heap size, try loading the file with more paralelism by decreasing split-size in TextInputFormat.SPLIT_MINSIZE and TextInputFormat.SPLIT_MAXSIZE (if you're using TextInputFormat) to elevate the level of paralelism.

Try using mapPartition instead of map so you can handle the computation inside a partition. If the computation uses a temporary variable or instance and you're still facing out of memory, try lowering the number of data per partition (increasing the partition number)

Job type: User-defined (StackOverflow)
Causes: Improper data partition (Expert)
Fix suggestions: increase the partition number by decreasing input split size. (suggestion)

6. Q: spark executor lost failure

User:
I am using the databricks spark cluster (AWS), and testing on my scala experiment. I have some issue when training on a 10 GB data with LogisticRegressionWithLBFGS algorithm. The code block where I met the issue is as follows:

First I got a lot executor lost failure and java out of memory issues, then I **repartitioned my training set with more partitions and the out of memory issues are gone,** but Still get executor lost failure.
Expert:

Job type: MLlib (StackOverflow)
Causes: Improper data partition (User)
Fix suggestions: add partition number (user fixed)

7. distinct on huge dataset

User:
I have a huge 2.5TB file. When i run:
val t = sc.textFile("/user/hdfs/dump.csv")
t.distinct.count
Got a bit further, i think out of memory error was caused by **setting spark.spill to false**. Now i have this error, is there an easy way to increase file limit for spark
Expert: You might try setting spark.shuffle.spill to false and see if that runs any longer (turning off shuffle spill is dangerous, though, as it may cause Spark to OOM if your reduce partitions are too large).

Job type: User-defined (Mailing list)
Causes: Improper data partition (us, reproduced)
Fix suggestions: set spark.spill = true, turning off shuffle spill is dangerous

8. Spark limitations question

User:
I'm currently testing a join of two data sets, Base and Skewed. They're both 100 million rows and they look like the following.
I have two tests:
1. join Base to itself, sum the "nums" and write out to HDFS
2. same as 1 except join Base to Skewed
(I realize the outputs are contrived and meaningless, but again, I'm testing limits here)

Test 2, however, works well on all but one of the nodes on the cluster. That node runs out of memory quickly and dies.
All of those nodes have 10 gigs of memory available to the spark executor and remaining
I'm assuming there's a build up of the **skewed 50million rows on the one particular node** and is running out of memory while it tries to merge them.

Expert:
Job type: User-defined (Mailing list)
Causes: Improper data partition (us, reproduced)
Fix suggestions: no

9. What should happen if we try to cache more data than the cluster can hold in memory?

User: If the RDD does not fit in memory, some partitions will not be cached and will be recomputed on the fly each time they're needed. This is the default level.

What I'm seeing per the discussion below is that when I try to cache more data than the cluster can hold in memory, I get:

Trying MEMORY_AND_DISK yields the same error.

Expert: It seems possible that you are running out of memory **unrolling a single partition of the RDD**. This is something that can cause your executor to OOM, especially if the cache is close to being full so the executor doesn't have much free memory left. How large are your executors? At the time of failure, is the cache already nearly full? A (potential) workaround would be to first persist your data to disk, then re-partition it, then cache it. I'm not 100% sure whether that will work though.

Job type: User-defined (Mailing list)
Causes: Improper data partition (Expert)
Fix suggestions: repartition (add partition number)

10. OOM writing out sorted RDD

User: Eventually, the logs show an OOM [1] included at the end of the mail followed by another TID loss to "FileSystem closed" errors indicated in stacktrace [2]. I noticed that **the partitions may be skewed as a result of the sort**, **with one or two paritions having upto 10% of all rows.** I also noticed that the data written out until the 50% stage (when memory shoots up) had a large number of empty part- files followed by a few files of 200M in size. While I could attempt to **partition manually (choosing custom ranges for a range partitioner**), it appears that limiting read sizes (from the earlier shuffle) during the write to HDFS should help successfully write out even overloaded partitions as well.
as expected, switching to kryo merely delays the inevitable. Does anyone have experience controlling memory consumption while processing (e.g. writing out) **imbalanced partitions**?

Job type: User-defined (Mailing list)
Causes: Improper data partition (User)
Fix suggestions: change partition function

11. Only master is really busy at KMeans training

User:
The training data is supplied in this form:
var vectors2 = vectors.repartition(1000).persist(org.apache.spark.storage.StorageLevel.MEMORY_AND_DISK_SER)
var broadcastVector = sc.broadcast(vectors2)

The 1000 partitions is something that could probably be optimized, but too few will cause OOM errors. As far as I can see, it's the repartitioning that is causing the problems. However, without that, I have only one partition for further RDD operations on dict, so it seems to be necessary.

Expert:

Job type: User-defined (Mailing list)
Causes: Improper data partition (User)
Fix suggestions: repartition (add partition number)

12. RDD Blocks skewing to just few executors

User:
As you can see most of the **data is skewing to just 2 executors, with 1 getting more than half the Blocks. These become a hotspot and eventually I start seeing OOM errors.** I've tried this a half a dozen times and the 'hot' executors changes, but not the skewing behavior.
Expert:

Job type: User-defined (Mailing list)
Causes: Improper data partition (User)
Fix suggestions: no

13. Using Spark on Data size larger than Memory size

User: Thank you. I'm using mapPartitions() but as you say, it requires that every partition fit in memory. This will work for now but may not always work so I was wondering about another way.
Expert: If an individual partition becomes too large to fit in memory then the usual approach would be to repartition to more partitions, so each one is smaller. Hopefully then it would fit. you can OOM under certain configurations, and it's possible you'll need to change from the default configuration if you're using doing very memory-intensive jobs. However, there are very few cases where Spark would simply fail as a matter of course -- for instance, you can always increase the number of partitions to decrease the size of any given one. or repartition data to eliminate skew.

Job type: User-defined (Mailing list)
Causes: Improper data partition (Expert)
Fix suggestions: repartition (add partition number, fixed)

4. Hotspot key

4.1 Hadoop Errors (2 errors here + 13 errors in Large accumulated results)

1. Q: Reducer's Heap out of memory

Expert: In some cases, JOIN (especially the replicated kind) will cause this type of issue. T**his happens when you have a "hot spot" of a particular key.** For example, say you are doing some sort of join and one of those keys shows up 50% of the time. Whatever reducer gets lucky to handle that key is going to get clobbered. You may want to investigate which keys are causing hot spots and handle them accordingly. In my data, usually these hot spots are useless anyways. To find out what's hot, just do a GROUP BY and COUNT and figure out what's showing up a lot. Then, if it's not useful, just FILTER it out.

Job type: User-defined (StackOverflow)

Causes: Hotspot key (Expert)
Fix suggestions: Filter the useless the hot key (Expert)
Fix details: Skip the hotspot records (fixed)

2. Case: Cogroup in Pig [page 75]

User/Expert: The next major concern is the possibility of **hot spots in the data that could result in an obscenely large record.** With large data sets, it is conceivable that a particular output record is going to have a lot of data associated with it. Imagine that for some reason a post on StackOverflow has a million comments associated with it. That would be extremely rare and unlikely, but not in the realm of the impossible.

Job type: Pig (Book, MapReduce Design Patterns)
Causes: Hotspot key (Expert)
Fix suggestions: no

4.2 Spark Errors (7 errors here + 1 error in Large accumulated results)

1. Q: Spark: out of memory exception caused by groupbykey operation

User：
I have a large file where each line is a record with an id as key.
so the lines with the same id will be shuffled to one worker node. a tuple in RDD groupedLines is like *id -> Iterable(line1, line2, ..., lineN)* if **lots of lines have the same id, then the size of the tuple's value *Iterable(...)* will be quite large**, and if it is larger then the JVM memory limit of the process on the machine, out of memory problem may happen.
Expert:
def groupByKey(numPartitions: Int): RDD[(K, Seq[V])]

Try to increase parameter

```
numPartitions
```

Job type: User-defined (StackOverflow)
Causes: Hotspot Key (User, Reproduced)
Fix suggestions: no

2. Partitioning - optimization or requirement?

User：
I'm often running out of memory when doing unbalanced joins (ie. **cardinality of some joined elements much larger than others**). Raising the memory ceiling fixes the problem, but that's a slippery slope.
Expert:
Job type: User-defined (Mailing list)
Causes: Hotspot Key (User, reproduced)
Fix suggestions: no

3. Lag function equivalent in an RDD

User：
Example:
val pairs = PositionPair.mapValues(kv=> List((kv._2.getTime, kv._2.getPos)))
//results an RDD with Int as Key and (Date, Position) as value.

The question: Is there a way to create a line segment for adjacent position for the same vehicle in a memory efficient way?

PS: I have tried reduceByKey and then splitting the List of position in tuples. For me **it runs out of memory every time because of the volume of data.**

Expert: You say you reduceByKey but are you really collecting all the tuples for a vehicle in a collection, like what groupByKey does already? Yes, **if one vehicle has a huge amount of data that could fail.**

Maybe you can consider using something like vehicle and *day* as a key. This would make you process each day of data separately, but if that's fine for you, might **drastically cut down the data associated to a single key**.

Job type: User-defined (Mailing list)
Causes: Hotspot Key (Expert)
Fix suggestions: change key (from vehicle to (vehicle, day)), cut down the data associated to a single key (Expert)

4. Understanding RDD.GroupBy OutOfMemory Exceptions

User：
I'm trying to do a simple groupByKey (see below), but it fails with a java.lang.OutOfMemoryError: GC overhead limit exceeded exception.
I can count the group sizes using reduceByKey without problems, ensuring myself the problem isn't caused by a single excessively large group, nor by an excessive amount of groups.
I've tried reformatting, reshuffling and increasing the groupBy level of parallelism:

```
 keyvals.groupByKey(24).count // fails
 keyvals.groupByKey(3000).count // fails
 keyvals.coalesce(24, true).groupByKey(24).count // fails
 keyvals.coalesce(3000, true).groupByKey(3000).count // fails
```

I've tried playing around with spark.default.parallelism, and increasing spark.shuffle.memoryFraction to 0.8 while lowering spark.storage.memoryFraction to 0.1.

Expert: The groupBy operator in Spark is not an aggregation operator (e.g. in SQL where you do select sum(salary) group by age...) - there are separate more efficient operators for aggregations. Currently groupBy requires that **all of the values for one key can fit in memory**. In your case, it's possible you have a single key with a very large number of values, given that your count seems to be failing on a single task. Within a partition things will spill - so the current documentation is correct. **This spilling can only occur *across keys* at the moment. Spilling cannot occur within a key at present**.

Job type: User-defined (Mailing list)
Causes: Hotspot Key (Expert, reproduced)
Fix suggestions: change key, sub-dividing any very large groups, aggregating partial values for each group (Expert)
Fix details: Typically approaches involve sub-dividing any very large groups, for instance, appending a hashed value in a small range (1-10) to large keys. Then your downstream code has to deal with aggregating partial values for each group. If your goal is just to lay each group out sequentially on disk on one big file, you can call `sortByKey` with a hashed suffix as well.

5. OutOfMemory in "cogroup"

User:There are too many values for a special key and these values cannot fit into memory. Spilling data to disk helps nothing because cogroup needs to read all values for a key into memory.
Expert: The problem is that the estimated size of used memory is inaccurate. I dig into the codes and found that SizeEstimator.visitArray randomly selects 100 cell and use them to estimate the memory size of the whole array. In our case, most of the biggest cells are not selected by SizeEstimator.visitArray, which causes the big gap between the

estimated size (27M) and the real one (5G). spark.shuffle.safetyFraction is useless for this case because the gap is really huge. Here 100 is controlled by SizeEstimator.ARRAY_SAMPLE_SIZE and there is no configuration to change it.

Job type: User-defined (Mailing list)
Causes: Hotspot Key (User, reproduced)
Fix suggestions: no

6. GroupBy Key and then sort values with the group

User: I am having a similar issue, but **I have a lot of data for a group & I cannot materialize the iterable into a List or Seq in memory.** [I tried & it runs into OOM]. is there any other way to do this ? I also tried a secondary-sort, with the key having the "group::time", but the problem with that is the same group-name ends up in multiple partitions & I am having to run sortByKey with one partition - sortByKey(true, 1) which shuffles a lot of data..

Expert: There is a new API called repartitionAndSortWithinPartitions() in master, it may help in this case, then you should do the `groupBy()` by yourself.

Job type: User-defined (Mailing list)
Causes: Hotspot Key (User, reproduced)
Fix suggestions: repartition
System requirement: disk-based data structure

7. Sorting partitions in Java

User: I'm trying to sort data in each partition of an RDD.
I was able to do it successfully in Scala like this:

```
val sorted = rdd.mapPartitions(iter => {
  iter.toArray.sortWith((x, y) => x._2.compare(y._2) < 0).iterator
},
preservesPartitioning = true)
```

I used the same technique as in OrderedRDDFunctions.scala, so I assume it's a reasonable way to do it.
I think the code I have will work for me, but I can imagine conditions where it will run out of memory

Expert: sortByKey currently requires partitions to fit in memory, but there are plans to add external sort, It's an Iterator in both Java and Scala. In both cases you need to copy the stream of values into something List-like to sort it. An Iterable would not change that (not sure the API can promise many iterations anyway).

Job type: User-defined (Mailing list)
Causes: Hotspot Key (us, reproduced)
Fix suggestions: no
System requirement: disk-based data structure

5. Large single record

5.1 Hadoop Errors (3 errors here + 3 errors in Large intermediate restuls)

1. Q: Hadoop Streaming Memory Usage

User: Reading the file from the HDFS and constructing a Text-Object should not amount to more than 700MB Heap - assuming that Text does also use 16-Bit per Character - I'm not sure about that but I could imagine that Text only uses 8-Bit.
So **there is these (worst-case) 700MB Line**. The Line should fit at least 2x in the Heap but I'm getting always out of memory errors.
Expert: Input File is a 350MByte file containg a single line full of a's.
I'm assuming you file has a single line of all a's with a single endline delimiter.
If that is taken up as a value in the map(key, value) function, I think, you might have memory issues, since, **you task have can use only 200MB and you have a record in memory which is of 350MB**.

Job type: User-defined (StackOverflow)
Causes: Large single record (Expert)
Fix suggestions: no

2. A: Hadoop Pipes: how to pass large data records to map/reduce tasks

User: I'm trying to use map/reduce to process large amounts of binary data. The application is characterized by the following: the number of records is potentially large, such that I don't really want to store each record as a separate file in HDFS (I was planning to concatenate them all into a single binary sequence file), and each record is a large coherent (i.e. non-splittable) blob, between one and several hundred MB in size. The records will be consumed and processed by a C++ executable. If it weren't for the size of the records, the Hadoop Pipes API would be fine: but this seems to be based around passing the input to map/reduce tasks as a contiguous block of bytes, which is impractical in this case.
Expert: Hadoop is not designed **for records about 100MB in size**. You will get OutOfMemoryError and uneven splits because some records are 1MB and some are 100MB. By Ahmdal's Law your parallelism will suffer greatly, reducing throughput.

Job type: User-defined (StackOverflow)
Causes: Large single record (Expert)
Fix suggestions: break up the large record into smaller records (Expert)
Fix details: Your first map task must break up the data into smaller records for further processing. Further tasks then operate on the smaller records. If you really can't break it up, make your map reduce job operate on file names. The first mapper gets some file names, runs them thorough your mapper C++ executable, stores them in more files. The reducer is given all the names of the output files, repeat with a reducer C++ executable. This will not run out of memory but it will be slow. (suggestion)

3. OutOfMemory Error

User: The key is of the form "ID :DenseVector Representation in mahout with dimensionality size = 160k"
For example: C1:[,0.00111111, 3.002, ... 1.001,....]
So, typical size of the key of the mapper output can be 160K*6 (assuming double in string is represented in 5 bytes)+ 5 (bytes for C1:[]) + size required to store that the object is of type Text
Yeah. That was the problem. And Hama can be surely useful for large scale matrix operations

Expert: I guess vector size seems too large so it'll need a distributed vector architecture (or 2d partitioning strategies) for large scale matrix operations.

Job type: User-defined (Mailing list)
Causes: Large single record (Expert)
Fix suggestions: no

5.2 Spark Errors (1)

1. OOM with groupBy + saveAsTextFile

User: I'm trying to run groupBy(function) followed by saveAsTextFile on an RDD of count ~ 100 million. The data size is 20GB and groupBy results in an RDD of 1061 keys with values being Iterable<Tuple4<String, Integer, Double, String>>. My understanding is that if each host is processing a single logical partition to saveAsTextFile and is reading from other hosts to write out the RDD, it is unlikely that it would run out of memory. My interpretation of the spark tuning guide is that the degree of parallelism has little impact in case (1) above since max cores = number of hosts. Can someone explain why there are still OOM's with 100G being available?

Expert: None of your tuning will help here because the problem is actually the way you are saving the output. If you take a look at the stacktrace, it is trying to **build a single string that is too large for the VM** to allocate memory. The VM is actually not running out of memory, but rather, JVM cannot support a single String so large. I suspect this is due to the fact that the **value in your key, value pair after group by is too long (maybe it concatenates every single record)**. Do you really want to save the key, value output this way using a text file? Maybe you can write them out as multiple strings rather than a single super giant string.

Job type: User-defined (Mailing list)
Causes: Large single record (Expert, reproduced)
Fix suggestions: write multiple times (split the single large record)
Fix details: Maybe you can write them out as multiple strings rather than a single super giant string (suggestion)

6. Large external data

6.1 Hadoop Errors (8)

1. Q: OutOfMemory Error when running the wikipedia bayes example on mahout

User: i ran mahout wikipedia example with the 7 gig wiki backup.. , but when testing the classifier, i am getting the a OutOfMemory Error
i have pasted the output below, i set the mahout heap size and java heap size to 2500m
Expert: You need to increase the memory available to mappers. Set mapred.map.java.child.opts to something big enough to hold the model.
It may be that you are trying to load something unrealistically large into memory. **he's running out of memory where the mapper side-loads a model.** It is not affected by how much overall data goes into the mapper.
Causes: Large external data (training model)

Job type: Apache Mahout (StackOverflow)
Causes: Large external data (large training model) (Expert, Reproduced)
Fix suggestions: Add memory space (Expert)

2. A: Hive: Whenever it fires a map reduce it gives me this error "Can not create a Path from an e...

User: I have solved the problem.
I looked up the log file and in my case **the table is an external table referring to a directory located on hdfs.** This directory contains more than 300000 files. So **while reading the files it was throwing an out of memory exception** and may be for this reason it was getting an empty string and throwing 'Can not create a Path from an empty string' exception.
I tried with a smaller subset of files and it worked.

Job type: Apache Hive (StackOverflow)

Causes: Large external data (large external table) (User)
Fix suggestions: decrease the dataset (User)
Fix details: I tried with a smaller subset of files and it worked

3. Case: Replicated Join [page 119]

User/Expert: A replicated join is an extremely useful, but has a strict size limit on all but one of the data sets to be joined. All the data sets except the very large one are essentially read into memory during the setup phase of each map task, which is limited by the JVM heap.

Job type: Pig join (Book, MapReduce Design Pattern)
Causes: Large external data (large table for join) (Expert, reproduced)
Fix suggestions: use reduce join (Expert)
Interesting: Yes

4. Case: ParserUserData [page 122]

User/Expert: During the setup phase of the mapper, the user data is read from the DistributedCache and stored in memory. Each record is parsed and the user ID is pulled out of the record. Then, t**he user ID and record are added to a HashMap for retrieval in the map method. This is where an out of memory error could occur**, as the entire contents of the file is stored, with additional overhead of the data structure itself. If it does, you will either have to increase the JVM size or use a plain reduce side join.

Job type: Pig join (Book, MapReduce Design Pattern)
Causes: Large external data (user data) (Expert)
Fix suggestions: no

5. Q: Mahout - Exception: Java Heap space

User: I'm trying to convert some texts to mahout sequence files using:
mahout seqdirectory -i Lastfm-ArtistTags2007 -o seqdirectory
But all I get is a OutOfMemoryError, as here:
Expert: add heap size

Job type: Apache Mahout (StackOverflow)
Causes: Large external data (large model) (us, reproduced)
Fix suggestions: no

6. Q: hadoop mapper over consumption of memory(heap)

User: I wrote a simple hash join program in hadoop map reduce. The idea is the following:
A small table is distributed to every mapper using DistributedCache provided by hadoop framework. The large table is distributed over the mappers with the split size being 64M. **The setup code of the mapper creates a hashmap reading every line from this small table. In the mapper code, every key is searched(get) on the hashmap, and if the key exists in the hash map it is written out.** There is no need of a reducer at this point of time. This is the code which we use:

While testing this code, **our small table was 32M, and large table was 128M**, one master and 2 slave nodes.
This code fails with the above inputs when I have a 256M of heap. I use -Xmx256m in the mapred.child.java.opts in mapred-site.xml file. When I increase it to 300m it proceeds very slowly and with 512m it reaches its max throughput.

Job type: User-defined (StackOverflow)
Causes: Large external data (external table) (us, reproduced)
Fix suggestions: no

7. Q: Mahout on Elastic MapReduce: Java Heap Space

User: I'm running Mahout 0.6 from the command line on an Amazon Elastic MapReduce cluster trying to canopy-cluster ~1500 short documents, and the jobs keep failing with a "Error: Java heap space" message.
Expert:
Job type: Apache Mahout (StackOverflow)
Causes: Large external data (external table) (us, reproduced)
Fix suggestions: no

8. OutOfMemoryError of PIG job (UDF loads big file)

User: I am running a hadoop job written in PIG. It fails from out of memory because a UDF function consumes a lot of memory, it loads a big file. What are the settings to avoid the following OutOfMemoryError?
Expert:

Job type: Apache Pig (Mailing list)
Causes: Large external data (User)
Fix suggestions: no

7. Large intermediate results

7.1 Hadoop Errors (4)

1. Q: java.lang.OutOfMemoryError on running Hadoop job

User: I have an input file (~31GB in size) containing consumer reviews about some products which I'm trying to lemmatize and find the corresponding lemma counts of. The approach is somewhat similar to the WordCount example provided with Hadoop. I've 4 classes in all to carry out the processing: StanfordLemmatizer [contains goodies for lemmatizing from Stanford's coreNLP package v3.3.0], WordCount [the driver], WordCountMapper [the mapper], and WordCountReducer [the reducer].
I've tested the program on a subset (in MB's) of the original dataset and it runs fine. Unfortunately, when I run the job on the complete dataset of size ~31GB, the job fails out.
Expert: You need to make the size of the individual units that you are processing (i.e., each Map job in the map-reduce) reasonable. The first unit is the size of document that you are providing to the StanfordCoreNLP's annotate() call. T**he whole of the piece of text that you provide here will be tokenized and processed in memory.** In tokenized and processed form, it is over an order of magnitude larger than its size on disk. So, the document size needs to be reasonable. E.g., you might pass in one consumer review at a time (and not a 31GB file of text!)
Secondly, **one level down, the POS tagger (which precedes the lemmatization) is annotating a sentence at a time, and it uses large temporary dynamic programming data structures for tagging a sentence, which might be 3 orders of magnitude larger in size than the sentence.** So, the length of individual sentences also needs to be reasonable. If there are long stretches of text or junk which doesn't divide into sentences, then you may also have problems at this level. One simple way to fix that is to use the pos.maxlen property to avoid POS tagging super long sentences.
Causes: Large intermediate results, it uses large temporary dynamic programming data structures for tagging a sentence, which might be 3 orders of magnitude larger in size than the sentence.

Job type: User-defined with third library (StackOverflow)
Causes: Large intermediate results + large single record (Expert & Reproduced)
Fix suggestions: split long records to multiple small records (Expert, accepted)
Fix details: the document size needs to be reasonable, avoid POS tagging super long sentences, If there are long stretches of text or junk which doesn't divide into sentences (suggestions)

2. Q: Writing a Hadoop Reducer which writes to a Stream

User: I have a Hadoop reducer which throws heap space errors while **trying to produce very long output records**. Is there a way to write a Reducer to use Streams for output, so that I can run through the data for **the record without marshalling the whole record into memory**?

Job type: User-defined (StackOverflow)
Causes: Large intermediate results (User)
Fix suggestions: no

3. Q: Hadoop Error: Java heap space

User: I am literally running an empty map and reduce job. However, the job does take in an input that is, roughly, about 100 gigs. For whatever reason, I run out of heap space. Although the job does nothing.
Note Got it figured out. Turns out I was setting the configuration to have a different terminating token/string. The format of the data had changed, so that token/string no longer existed. **So it was trying to send all 100gigs into ram for one key.**

Job type: User-defined (StackOverflow)
Causes: Large intermediate results + Large single record (User)
Fix suggestions: Split the large single record
Fix details: Note Got it figured out. Turns out I was setting the configuration to have a different terminating token/string. The format of the data had changed, so that token/string no longer existed. So it was trying to send all 100gigs into ram for one key (fixed)

4. Q: Heap error when using custom RecordReader with large file

User: I've written a custom file reader to not split my input files as they are large gzipped files and I want my first mapper job to simply gunzip them. I followed the example in 'Hadoop The Definitive Guide' but I get a heap error when trying to read in to the BytesWritable. I believe this is because the byte array is of size 85713669, but I'm not sure how to overcome this issue.
Expert: In general you **can not load whole file into memory** of Java VM. You should find some streaming solution to process large files - read data chunk by chunk and save the results w/o fixing in memory whole data set.
Cause: Large intermediate results
Large bye[] in the RecordRead. In other words, large intermediate results are generated for each input record.

Job type: User-defined (StackOverflow)
Causes: Large intermediate results + Large single record (Expert)
Fix suggestions: read data chunk by chunk, avoid load the whole file (Expert)
Fix details: In general you can not load whole file into memory of Java VM. You should find some *streaming solution* to process large files - read data chunk by chunk and save the results w/o fixing in memory whole data set
This specific task - unzip is probably not suited for the MR since there is no logical division of data into records. (suggestion)

7.2 Spark Errors (2)

1. OutOfMemory Error

User: I am trying to implement machine learning algorithms on Spark. I am working on a 3 node cluster, with each node having 5GB of memory. Whenever I am working with slightly more number of records, I end up with OutOfMemory Error. Problem is, even if **number of records is slightly high, the intermediate result from a transformation is huge and this results in OutOfMemory Error.** To overcome this, we are partitioning the data such that each partition has only a few records.

Expert: Increase the number of partitions 3. You could try persisting the RDD to use DISK_ONLY. If increasing the partitions is the only the way, then one might end up with OutOfMemory Errors, when working with certain algorithms where intermediate result is huge.
Job type: User-defined (Mailing list)
Causes: Large intermediate results (User)
Fix suggestions: add partition number, change storage level

2. MLLib ALS question

User: I'm trying to use Matrix Factorization over a dataset with like 6.5M users, 2.5M products and 120M ratings over products. The test is done in standalone mode, with unique worker (Quad-core and 16 Gb RAM).
The program runs out of memory, and I think that this happens because flatMap holds data in memory.
(I tried with Movielens dataset that has 65k users, 11k movies and 100M ratings and the test does it without any problem)

Expert: The current ALS implementation constructs all subproblems in memory. With rank=10, that means (6.5M + 2.5M) * 10^2 / 2 * 8 bytes = 3.5GB. The ratings need 2GB, not counting the overhead. ALS creates in/out blocks to optimize the computation, which takes about twice as much as the original dataset. Note that this optimization becomes "overhead" on a single machine. All these factors contribute to the OOM error. ALS still needs to load and deserialize the in/out blocks (one by one)
from disk and then construct least squares subproblems. All happen in RAM. The final model is also stored in memory.

Job type: MLlib (Mailing list)
Causes: Large intermediate results (Expert, reproduced)
Fix suggestions: I added the ALS.setIntermediateRDDStorageLevel and it worked well (a little slow, but still did the job and i've made MF and get all the features).
But even if I persist with DISK_ONLY, the system monitor shows on the memory and swap history that Apache Spark is using RAM. (fixed)

8. Large accumulated results

8.1 Hadoop Errors (30)

1. Q: Getting java heap space error while running a mapreduce code for large dataset

User: am a beginner of MapReduce programming and have coded the following Java program for running in a Hadoop cluster comprising 1 NameNode and 3 DatanNodes :
Each row has an ID followed by 3 comma-separated attributes. My problem is to find the frequency of the value of each attribute(along the column not across the row if the dataset is seen as a 2-D array) of each ID and then sum up the frequencies of each attribute for an ID and find the average.Thus for the above the dataset:
Expert: In your first job you are keeping all the values corresponding to a specific key in a list. As you have 5cr rows and each row have 9 attributes the size of all the values corresponding to a specific key will be too large for a normal List in java to keep in heap memory.That is the reason for java.lang.OutOfMemoryError: Java heap space exception.You have to avoid keeping all the values correponding to a key in an object in java heap.
Causes: Large accumulated results, hotspot key.

Job type: User-defined (StackOverflow)
Causes: Large accumulated results + Hotspot key (Expert, Reproduced)
Fix suggestions: avoid accumulation (Expert)
Fix details: You have to avoid keeping all the values correponding to a key in an object in java heap.
System requirement: memory+disk data structures

2. Q: A join operation using Hadoop MapReduce

User: This solution is very good and works for majority of the cases but in my case my issue is rather different. **I am dealing with a data which has got billions of records and taking a cross product of two sets is impossible because in many cases the hashmap will end up having few million objects.** So I encounter a Heap Space Error.
Expert: You should look into how Pig does skew joins. The idea is that if **your data contains too many values with the same key (even if there is no data skew)**, you can create artificial keys and spread the key distribution. This would make sure that each reducer gets less number of records than otherwise. For e.g. if you were to suffix "1" to 50% of your key "K1" and "2" the other 50% you will end with half the records on the reducer one (1K1) and the other half goes to 2K2.

Job type: User-defined (StackOverflow)
Causes: Large accumulated results + Hotspot key (User and Expert, reproduced)
Fix suggestions: Redesign the key, use "skew join" (Expert) create artificial keys and spread the key distribution
Fix details: Redesign the key: suffix "1" to 50% of your key "K1" and "2" the other 50%. (create artificial keys and spread the key distribution), could some kind of sampling algorithm. (suggestion)

3. Q: Detailed dataflow in hadoop's mapreduce?

User: I am struggling a bit to understand the dataflow in mapreduce. Recently a very demanding job crashed when my disks ran out of memory in the reduce phase. I find it difficult to estimate how much disk my job will need. I will describe the dataflow in detail. My map tasks are very similar to wordcount, so they need little memory. **My reduce tasks work with permutation groups of words. Since identical words need to be joined the reduce function requires a temporary hash map which is always <= 3GB**.

Job type: User-defined (StackOverflow)
Causes: Large accumulated results (User)
Fix suggestions: no

4. Q: Building Inverted Index exceed the Java Heap Size

User: I am building an inverted index for large data set (One day worth of data from large system). The building of inverted index get executed as a map reduce job on Hadoop. Inverted index is build with the help of scala. Structure of the inverted index as follows: {key:"New", ProductID:[1,2,3,4,5,...]} these get written in to avro files.
During this process I run into Java Heap Size issue. I think the reason is that terms like "New" as I showed above contain large number of productId(s). I have a, rough idea where the issue could take place in my Scala code:
When I execute the map reduce job for small data set I don't get the error. Which means that as the data increase number of items/product_ids that I index for words like New or old etc get bigger which cause the Heap Size to overflow. So, the question is how can avoid java heap size overflow and accomplish this task.
I don't have a exact number it varies because some key words only return 1 or 2 items but some may return 100000

Expert: you might have to do the operation **in several passes so you don't accumulate too much data in memory** (it looks to me like you have all product IDs for all keys in memory at the same time).

Job type: User-defined (StackOverflow)
Causes: Large accumulated results + Hotspot key (Expert, Reproduced)
Fix suggestions: Job split, do the operation in several passes (Expert)
Fix details: do the operation in several passes (suggestion)

5. Q: Out of memory due to hash maps used in map-side aggregation

User: MY Hive Query is throwing this exception.

I tried this on 8 EMR core instances with 1 large master instance on 8Gb of data. First i tried with external table (location of data is path of s3). After that i downloaded data from S3 to EMR and used native hive tables. But in both of them i got the same error
yes, my mapper is sorting.......i tried to debug the problem and it comes out that my mapper was using 5.5 gb of RAM. So, i increased the RAM and it worked. Before increasing the RAM i also tried set hive.map.aggr.hash.percentmemory = 0.25. But it didn't work.
Expert: Out of memory due to hash maps used in map-side aggregation. if mapper is not sorting: consider bumping the hash aggregate memory % as indicated in the log to a higher number. if mapper is sorting - just bump up task memory to a larger number.

Job type: Apache Hive (StackOverflow)
Causes: Large accumulated results (Expert, Reproduced)
Fix suggestions: lower hive.map.aggr.hash.percentmemory (Expert)

6. Q: Reducer's Heap out of memory

Expert: Another source of this problem is a Java UDF that is aggregating way too much data. For example, if you have a UDF that goes through a data bag and collects the records into some sort of list data structure, you may be blowing your memory with a hot spot value.

Job type: Pig UDF (StackOverflow)
Causes: Large accumulated results (Expert)
Fix suggestions: no

7. Q: Hadoop UniqValueCount Map and Aggregate Reducer for Large Dataset (1 billion records)

User: a set that has approximately 1 billion data points. There are about 46 million unique data points I want to extract from this.
I want to use Hadoop to extract the unique values, but keep getting "Out of Memory" and Java heap size errors on Hadoop - at the same time, I am able to run this fairly easily on a single box using a Python Set (hashtable, if you will.) and then running the "aggregate" reducer to get the results, which should look like this for the above data set:
Expert: You're probably getting the memory error in the shuffle, remember that Hadoop sorts the keys before starting the reducers. Sort itself is not necessary for most apps, but Hadoop uses this as a way to aggregate all value belonging to a key. For your example, your mappers will end up writing a lot of times the same values, while you only care about **how many uniques you have for a given key**. Add a combiner in your job that will run just after the mapper but before the reducer, and will only keep uniques before sending to the reducer. Modify your mapper to keep a mapping of what you already sent, and not send if you've already sent this mapping before.

Job type: User-defined (StackOverflow)
Causes: Large accumulated results + Hotspot key (Expert)
Fix suggestions: add combiner to reduce the values for a key, in-memory combing (Expert)
Fix details: Add a combiner in your job that will run just after the mapper but before the reducer.

8. Q: Mahout Canopy Clustering, K-means Clustering : Java Heap Space - out of memory

User: More then a certain number of canopy centers, **the jobs keep failing with a "Error: Java heap space" message at 67% of reduce phase.**
K-means clustering also has same heap space problems, if the value of K is increasing.
I've heard canopy-center vectors and k-center vectors are held in memory on every mapper and reducer. That would be num of canopy center(or k) x sparsevector(300000 size) = enough for 4g memory things, which doesn't seem too bad.

Job type: Apache Mahout (StackOverflow)
Causes: Large accumulated results (User)
Fix suggestions: no (Enlarge memory limit cannot work)

9. Q: HBase: How to handle large query results

User: In my current approach, my function scans for the records and gets them just fine. I then try to convert the ResultScanner contents into ArrayList form so I can pass them to the calling function. I could just pass the Scanner, but I want to close it.
My problem is when the **ArrayList gets filled to about 2 million records**, I get an OutOfMemoryError:

Job type: User-defined (StackOverflow)
Causes: Large accumulated results (User)
Fix suggestions: no
System requirement: disk-based data structure

10. Q: OOM exception in Hadoop Reduce child

User: I am getting OOM exception (Java heap space) for reduce child. In the reducer, I am appending all the values to a StringBuilder which would be the output of the reducer process. The number of values aren't that many.
I had a set accumulating all the values - so that there are no duplicate values. I removed it later on.
Some sample sizes of iterator are as follows: 238695, 1, 13, 673, 1, 1 etc. These are not very large values. Why do I keep getting the OOM exception? Any help would be valuable to me.
Expert: 238695 seems pretty excessive - what's the typical length of the values being accumulated? If you really want to do this, and not get the OOM error then you'll need to look into a custom output format so you don't accumulate the values in memory
So for your example, you want to output the values for a particular key as a space separated list of the values (as the output key), and an empty text as the output value.

Job type: User-defined (StackOverflow)
Causes: Large accumulated results + Hotspot key (Expert, Reproduced)
Fix suggestions: multiple outputs (Expert) (Enlarge memory limit cannot work)

11. Subject: java.lang.OutOfMemoryError when using TOP udf

User: The immediate cause of the problem I had (failing without erroring out) was slightly different formatting between the small and large input sets. Duh. When I fixed that, I did indeed get OOM due to the nested distinct. I tried the workaround you suggested Jonathan using two groups, and it worked great! In a separate run I also tried SET pig.exec.nocombiner true; and found that worked also, and the runtime was the same as using the two group circumlocution.
Expert: I took a look and it is being reasonable. I do that that that is the issue: the way that **it works is by holding a priority queue of however many items you care about, adding one, then popping the bottom one. If it has to hold almost 3M objects in memory, memory issues is a real likely thing.** A couple things you can do: - have fewer columns. ie only do "TOP" of the things you really care about - more memory (don't you love that?)

Job type: Pig script (Mailing list)
Causes: Large accumulated results (Expert)
Fix suggestions: Filter the unrelated columns (TOP)
Fix details: have fewer columns. ie only do "TOP" of the things you really care about

12. Subject: java.lang.OutOfMemoryError while running Pig Job

User: I am running following Pig script in Pig 0.8 version
filter_pe = FILTER page_events BY is_iab_robot == 0 AND is_pattern_match_robot == 0 AND day == '2011-05-10';
select_pe_col = FOREACH filter_pe GENERATE day, is_cookied_user, anon_cookie, reg_cookie, referrer_id, has_web_search_phrase, business_unit_id; select_ref_col = FOREACH referrer_group_map GENERATE referrer_id,

has_web_search_phrase, referral_type_id; jn = JOIN select_ref_col BY (referrer_id, has_web_search_phrase), select_pe_col BY (referrer_id, has_web_search_phrase);
Expert: The stack trace shows that the OOM error is happening when the distinct is being applied. It looks like in some record(s) of the relation group_it, one more of the following bags is very large - logic.c_users, logic.nc_users or logic.registered_users;

Job type: Pig script (Mailing list)
Causes: Large accumulated results (Expert, Reproduced)
Fix suggestions: Try setting the property pig.cachedbag.memusage to 0.1 or lower (-Dpig.cachedbag.memusage=0.1 on java command line). It controls the memory used by pig internal bags, including those used by distinct. If that does not work, you can try computing count-distinct for each type of user separately and then combining the result. (suggestion)

13. Subject: Set number Reducer per machines.

User: I was using the following file for mapreduce job.
Cloud9/src/dist/edu/umd/cloud9/example/cooccur/ComputeCooccurrenceMatrixStripes.java
I am trying to run a job on my hadoop cluster, where I get consistently get heap space error. I increased the heap-space to 4 GB in hadoop-env.sh and reboot the cluster. However, I still get the heap space error.
Expert: It looks like both the mapper and reducer are using a map structure which will be created on the heap. **All the values from the reducer are being inserted into the map structure. If you have lots of values for a single key then you're going to run out of heap memory really fast.** Do you have a rough estimate for the number of values per key? We had this problem when we first started using map-reduce (we'd create large arrays in the reducer to hold data to sort). Turns out this is generally a very bad idea (it's particularly bad when the number of values per key is not bounded since sometimes you're algorithm will work and other times you'll get out of memory errors). In our case we redesigned our algorithm to not require holding lots of values in memory by taking advantage of Hadoop's sorting capability and secondary sorting capability.

Job type: Cloud9 (Mailing list)
Causes: Large accumulated results (Expert, Reproduced)
Fix suggestions: Change sort by using Hadoop secondary sort (Expert)
Fix details: redesigned our algorithm to not require holding lots of values in memory by taking advantage of Hadoop's sorting capability and secondary sorting capability (fixed)

14. Subject: ORDER ... LIMIT failing on large data

User: I have a small pig script that outputs the top 500 of a simple computed relation. It works fine on a small data set but fails on a larger (45 GB) data set. I don't see errors in the hadoop logs (but I may be looking in the wrong places). On the large data set the pig log shows. When I fixed that, I did indeed get OOM due to the nested distinct. I tried the workaround
you suggested Jonathan using two groups, and it worked great!
Expert: Nested distincts are dangerous. They are not done in a distributed fashion, they have to be loaded into memory. So that is what is killing it, not the order/limit. The alternative is to do two groups, first group by (citeddocid,CitingDocids) to get the distinct and then by citeddocid to get the count

Job type: Pig script (Mailing list)
Causes: Large accumulated results (Expert, Reproduced)
Fix suggestions: two groups (Expert)
Fix details: use two groups (fixed)

15. Subject: MapReduce Algorithm - in Map Combining

User/Expert: One common solution to limiting memory usage when using the in-mapper combining technique is to "block" input key-value pairs and "flush" in-memory data structures periodically. The idea is simple: instead of emitting intermediate data only after every key-value pair has been processed, emit partial results after processing every n key-value pairs. This is straightforwardly implemented with a counter variable that keeps track of the number of input key-value pairs that have been processed. As an alternative, **the mapper could keep track of its own memory footprint and flush intermediate key-value pairs once memory usage has crossed a certain threshold**. In both approaches, either the block size or the memory usage threshold needs to be determined empirically: with too large a value, the mapper may run out of memory, but with too small a value, opportunities for local aggregation may be lost.

Job type: User-defined (developer's blog)
Causes: Large accumulated results (User, Reproduced)
Fix suggestions: Map aggregation => Reduce aggregation, avoid in-memory aggregation (us)
Fix details: spill the accumulated results periodically or achieve a certain threshold. Emit partial results after processing every n key-value pairs

16. Subject: Efficient Sharded Positional Indexer

User/Expert: For frequent terms such as "the", the reducer output record may exceed the memory limit of the JVM, resulting in out of memory error. This is because Hadoop keeps the whole record (in this case the whole postings list for "the") in memory before sending it to disk. Partitioning the collection into more shards helps, but it's a suboptimal hack. **One way to avoid such error is to partition these large postings into manageable sized chunks, and output several records for the same key** (the word "the"). E.g. record1: <key="the", value=<<"doc1", tf, positions>, <"doc2", tf, pos> ...<"doc1000", tf, pos>>>, record2: <key="the", value=<<"doc1001", tf, pos>, <"doc1002", tf, pos> ...<"doc2000", tf, pos>>>, ..., recordN.

Job type: User-defined (developer's blog)
Causes: Large accumulated results + Hotspot key (User, Reproduced)
Fix suggestions: output several records for the same key (us)
Fix details: spill the accumulated results periodically. Output several records (e.g., 1000) for the same key

17. Case: Median and standard deviation [page 25]

User/Expert: The easiest way to perform these operations involves copying the list of values into a temporary list in order to find the median or iterating over the set again to determine the standard deviation. With large data sets, this implementation may result in Java heap space issues, because each value is copied into memory for every input group. Any sort of memory growth in the reducer has the possibility of blowing through the Java virtual machine's memory. For example, **if you are collecting all of the values into an ArrayList to perform a median, that ArrayList can get very big. This will not be a particular problem if you're really looking for the top ten items, but if you want to extract a very large number you may run into memory limits.**

Job type: User-defined (book, MapReduce Design Patterns)
Causes: Large accumulated results (Expert, Reproduced)
Fix suggestions: no
Interesting: Yes

18. Case: Sort XML Object [page 75]

User/Expert: If you are building some sort of XML object, all of those comments at one point might be stored in memory before writing the object out. This can cause you to blow out the heap of the Java Virtual Machine, which obviously should be avoided.

Job type: User-defined (book, MapReduce Design Patterns)
Causes: Large accumulated results (Expert)
Fix suggestions: no

19. Case: Reduce-side Join [page 66]

User/Expert: All the tuples from S with the same join key will be encountered first, which the reducer can buffer in memory. As the reducer processes each tuple from T, it is crossed with all the tuples from S. Of course, we are assuming that the tuples from S (with the same join key) will fit into memory, which is a limitation of this algorithm (and why we want to control the sort order so that the smaller dataset comes first).

Job type: User-defined (book, Data-Intensive Text Processing with MapReduce)
Causes: Large accumulated results (Expert)
Fix suggestions: control the sort order (Expert) (tricky method)

20. Case: BuildInvertedIndex [page 77]

User/Expert: The inverted indexing algorithm presented in the previous section serves as a reasonable baseline. However, there is a significant scalability bottleneck: the algorithm assumes that there is sufficient memory to hold all postings associated with the same term. **Since the basic MapReduce execution framework makes no guarantees about the ordering of values associated with the same key, the reducer first buffers all postings (line 5 of the reducer pseudo-code in Figure 4.2) and then performs an in-memory sort before writing the postings to disk.7 For efficient retrieval, postings need to be sorted by document id.** However, as collections become larger, postings lists grow longer, and at some point in time, reducers will run out of memory.

inverted indexing is nothing but a very large distributed sort and group by operation! We began with a baseline implementation of an inverted indexing algorithm, but quickly noticed a scalability bottleneck that stemmed from having to buffer postings in memory. Application of the value-to-key conversion design pattern (Section 3.4) addressed the issue by **offloading the task of sorting postings by document id to the MapReduce execution framework.**

A two-pass solution that involves first buffering the postings (in memory) would suffer from the memory bottleneck we've been trying to avoid in the first place.

There is a simple solution to this problem. Since the execution framework guarantees that keys arrive at each reducer in sorted order, one way to overcome the scalability bottleneck is to let the MapReduce runtime do the sorting for us. Instead of emitting key-value pairs of the following type:

Job type: User-defined (book, Data-Intensive Text Processing with MapReduce)
Causes: Large accumulated results (Expert)
Fix suggestions: Use framework sort mechanism (Expert) (fixed)
Interesting: Yes

21. Case: One-to-man join [page 65]

User/Expert: one-to-many join. Assume that tuples in S have unique join keys (i.e., k is the primary key in S), so that S is the “one” and T is the “many”. The above algorithm will still work, but when processing each key in the reducer, we have no idea when the value corresponding to the tuple from S will be encountered, since values are arbitrarily ordered. The easiest solution is to buffer all values in memory, pick out the tuple from S, and then cross it with every tuple from T to perform the join. However, as we have seen several times already, this creates a scalability bottleneck since we may not have sufficient memory to hold all the tuples with the same join key.

Job type: User-defined (book, Data-Intensive Text Processing with MapReduce)
Causes: Large accumulated results (Expert)

Fix suggestions: no

22. Case: word co-occurrence [page 59]

User/Expert: The computation of the word co-occurrence matrix is quite simple if the entire matrix fits into memory—however, in the case where the matrix is too big to fit in memory, a naı̈ve implementation on a single machine can be very slow as memory is paged to disk. Although compression techniques can increase the size of corpora for which word co-occurrence matrices can be constructed on a single machine, it is clear that there are inherent scalability limitations. We describe two MapReduce algorithms for this task that can scale to large corpora.

This algorithm will indeed work, but it suffers from the same drawback as the stripes approach: as the size of the corpus grows, so does that vocabulary size, and at some point **there will not be sufficient memory to store all co-occurring words and their counts for the word we are conditioning on.**

The insight lies in properly sequencing data presented to the reducer. If it were possible to somehow compute (or otherwise obtain access to) the marginal in the reducer before processing the joint counts, the reducer could simply divide the joint counts by the marginal to compute the relative frequencies.

Job type: User-defined (book, Data-Intensive Text Processing with MapReduce)
Causes: Large accumulated results (Expert)
Fix suggestions: no

23. Case: SortByKey [page 61]

User/Expert: However, since MapReduce makes no guarantees about the ordering of values associated with the same key, the sensor readings will not likely be in temporal order. The most obvious solution is to buffer all the readings in memory and then sort by timestamp before additional processing. However, it should be apparent by now that any in-memory buffering of data introduces a potential scalability bottleneck. What if we are working with a high frequency sensor or sensor readings over a long period of time? What if the sensor readings themselves are large complex objects? This approach may not scale in these cases—the reducer would run out of memory trying to buffer all values associated with the same key.

Job type: User-defined (book, Data-Intensive Text Processing with MapReduce)
Causes: Large accumulated results + Hotspot key(Expert)
Fix suggestions: no

24. Case: In-memory combining [page 54]

User/Expert: For both algorithms, the in-mapper combining optimization discussed in the previous section can also be applied; the modification is sufficiently straightforward that we leave the implementation as an exercise for the reader. However, the above caveats remain: there will be far fewer opportunities for partial aggregation in the pairs approach due to the sparsity of the intermediate key space. The sparsity of the key space also limits the effectiveness of in-memory combining, since the mapper may run out of memory to store partial counts before all documents are processed, necessitating some mechanism to periodically emit key-value pairs (which further limits opportunities to perform partial aggregation). Similarly, for the stripes approach, memory management will also be more complex than in the simple word count example. **For common terms, the associative array may grow to be quite large**, necessitating some mechanism to periodically flush in-memory structures.

Job type: User-defined (book, Data-Intensive Text Processing with MapReduce)
Causes: Large accumulated results + Hotspot key(Expert)
Fix suggestions: no

25. Q: Fail to join large groups

I have two data structures with the same structure:

{(id, (record1, record2, record3))}
I want to join them, by on the value of record1. In order to do that I wrote this script:

```
data_1_group = group data_1 by $1.record1;
data_2_group = group data_2 by $1.record1;
jj = join data_1_group by group, data_2_group by group;
```

But, since the both data_1 and data_2 contains millions of records while record1 can assume only 20 different values, the groups are very large and the script runs out of memory and fails.

Job type: Pig (StackOverflow)
Causes: Large accumulated results + Hotspot key (us, reproduced)
Fix suggestions: no

26. Hashing two relations

User: I know that map-side join does this (on pre-partitioned data), but I want to do it on reduce side. Using job-chaining, I can output (hash(key), value) by two map tasks on the two input files, but when it comes to the reduce stage, i have to take the same partition from both the hash tables. I am not sure how can I accomplish this. get a partition from each
Expert: If you want to do the join in reduce side, MapReduce framework enable this by grouping all the matching tuples together. Why bother to build hash table to buffer the entire partition in memory? This probably brings you a out-of-memory error. The default reduce join should be your choice in this case

Job type: User-defined (Mailing list)
Causes: Large accumulated results (Expert)
Fix suggestions: no

27. OutOfMemory during Plain Java MapReduce

User: during the Reduce phase or afterwards (i don't really know how to debug it) I get a heap out of Memory Exception.
I guess this is because the value of the reduce task (a Custom Writable) holds a List with a lot of user ids.
I had a look to the stacktrace and it says the problem is at the reducer:
userSet.add(iterator.next().toString());

Expert: When you implement code that starts memory-storing value copies for every record (even if of just a single key), things are going to break in big-data-land. Practically, post-partitioning What if we are working with a high frequency sensor or sensor readings over a long period of time? if you really really want to do this, or use an alternative form of key-value storage where updates can be made incrementally (Apache HBase is such a store, as one example).

This has been discussed before IIRC, and if the goal were to store the outputs onto a file then its better to just **directly serialize them with a file opened instead of keeping it in a data structure and serializing it at the end**. The caveats that'd apply if you were to open your own file from a task are described at

Looking at your reducer code, it appears that you are trying to compute the distinct set of user IDs for a given reduce key. Rather than computing this by holding the set in memory, **use a secondary sort of the reduce values**, then while iterating over the reduce values, look for changes of user id. Whenever it changes, write out the key and the newly found value.

Job type: User-defined (Mailing list)
Causes: Large accumulated results + Hotspot Key (Expert, reproduced)
Fix suggestions: Write the computing results continuously, user a secondary sort (Expert)
Fix details: You'd probably need to write out something continuously if you really really want to do this, or **use an alternative form of key-value storage where updates can be made incrementally** (Apache HBase is such a store, as one example).

28. how to solve reducer memory problem?

User: I have a map reduce program that do some matrix operations. in the reducer, it will average many large matrix(each matrix takes up 400+MB(said by Map output bytes). so if there 50 matrix to a reducer, then the total memory usage is 20GB. so the reduce task got exception
one method I can come up with is use Combiner to save sums of some matrixs and their count but it still can solve the problem because the combiner is not fully controled by me.
Expert: In your implementation, you Could OOM as you **store more and more data into "TrainingWeights result"**. So the question is for each "Reducer group", or "Key", how many data it could be?
If a key could contain big values, then all these values will be saved in the memory of "result" instance. That will require big memory. If so, either you have to have that much memory, or redesign your key, make it more lower level, so requires less memory
Job type: User-defined (Mailing list)
Causes: Large accumulated results + Hotspot Key (Expert)
Fix suggestions: Use combiner, redesign the key (User) (suggestions)

29. memoryjava.lang.OutOfMemoryError related with number of reducer?

User: I can fix this by changing heap size. But what confuse me is that when i change the reducer number from 24 to 84, there's no this error. Seems i found the data that causes the error, but i still don't know the exactly reason.
I just do a group with pig latin:
The group key (custid, domain, level, device) is significantly skewed, about 42% (58,621,533 / 138,455,355) of the records are the same key, and only the reducer which handle this key failed.

Expert: When you increase the number of reducers they each have less to work with provided the data is distributed evenly between them - in this case about one third of the original work. It is essentially the same thing as increasing the heap size - it's
just distributed between more reducers.

It is because of the nested distinct operation relies on the RAM to calculate unique values.

Job type: Apache Pig (Mailing list)
Causes: Large accumulated results - Count(distinct) + Hotspot Key (Expert)
Fix suggestions: Add reduce number (User)
Fix details: increase reducer number

30. Q: Why does the last reducer stop with java heap error during merge step

User: I keep increasing the number of reducers and I see that while all except one reducers run quickly and finish their job, one last reducer just hangs at the merge step with this message in its tasktracker log:
Yes, you are right. So I am wondering what is the solution for this.

Expert: My guess is that you have a single key with a huge number of values and the following line in your reducer is causing you problems: You can probably confirm this by putting in some debug and inspecting the logs for the reducer that never ends:

Job type: User-defined (StackOverflow)
Causes: Large accumulated results + Hotspot Key (Expert)
Fix suggestions: no

8.2 Spark Errors (10)

1. Q: Memory issues when running Spark job on relatively large input

User:
Particularly allocating a buffer of size 40M for each file in order to read the content of the file using BufferedInputStream. This causes the stack memory to end at some point.

The thing is:

- If I read line by line (which does not require a buffer), it will be very non-efficient read
- If I allocate one buffer and reuse it for each file read - is it possible in parallelism sense? Or will it get overwritten by several threads?

Expert:
It seems like you are reading the content of all input files into an in-memory ArrayList? This sort of defeats the purpose of working with RDDs/partitions,

Job type: User-defined (StackOverflow)
Causes: Large accumulated results (Expert, Reproduced)
Fix suggestions: moving the byte array allocation outside the iterator, so it gets reused by all partition elements
System requirement: disk-based data structure

2. Q: spark aggregatebykey with collection as zerovalue

User:
I'm working with a rdd of tuples [k, v(date, label)] and I'm trying to get all the distinct labels and the min of date for each keys.

I've ended with this piece of code :`aggregateByKey((new DateTime(), new mutable.HashSet[String]()))((acc: (DateTime, mutable.HashSet[String]), v: (DateTime, String)) => (if (acc._1.isBefore(v._1)) acc._1 else v._1, acc._2 + v._2), (acc1: (DateTime, mutable.HashSet[String]), acc2: (DateTime, mutable.HashSet[String])) => (if (acc1._1.isBefore(acc2._1)) acc1._1 else acc2._1, acc1._2 ++ acc2._2))`

Expert:
Ok, there are many things going on here, so let's go one by one:

1. groupByKey will just shuffle all data for a key to a single executor, load it into memory and make it available for you to do whatever you want (aggregation or not). This is an immediate cause of possible OutOfMemoryErrors if there is a lot of data associated with any given key (skewed data).

2. aggregateByKey will try to be smarter. Since it knows that is aggregating, it will try to aggregate locally before shuffling anything. The methods and zero-value you provide are serialize to multiple executors in order to

accomplish just this. So your aggregation logic will be distributed even for the same key. Only accumulators will be serialized and merged. So overall, this method is significantly better in most cases, but you have to be careful still if (like in this case) the size of the accumulator itself can grow without bounds. Relevant questions: How many strings you expect per key? How big are these strings? How much de-duplication you expect to happen?

3. Another thing you can do is to take this piece of advice from aggregateByKey's documentation:

Job type: User-defined (StackOverflow)
Causes: Large accumulated results (Expert)
Fix suggestions: increase the number of partitions, change groupByKey() to aggregateByKey (aggregate locally before shuffling anything, your aggregation logic will be distributed even for the same key)

3. Q: Spark: Out Of Memory Error when I save to HDFS

User:
I am experiencing OOME when I save big data to hdfs

```
val accumulableCollection = sc.accumulableCollection(ArrayBuffer[String]()) val rdd = textfile.filter(row => {
        if (row.endsWith(",")) {
        accumulableCollection += row
        false
} else if (row.length < 100) {
        accumulableCollection += row
        false
}
        valid
})
```

the accumulableCollection that will be written in HDFS has the max size of 840MB or 1.3M rows. in this scenario I am just writing 146MB of data. yes, the accumulableCollection that will be written in HDFS has the max size of 840MB or 1.3M rows. in this scenario I am just writing 146MB of data.
Expert:
It means pretty much what it says. You are trying to serialize a single object which is very large. You should probably rewrite your code to not do this.

Job type: User-defined (StackOverflow)
Causes: Large accumulated in accumulableCollection (Expert, reproduced)
Fix suggestions: no

4. Common crawl parsing has high fan out and runs out of memory

User: I am trying to parse the commoncrawl.org data, which is stored in amazon s3 as a series of (not so) large (1G) gzip files. My goal is to call flatMap which will receive a gzip file as input and yield a few dozen thousands html blobs as result. See below the code with some boilerplate stripped.

```
WarcReaderCompressed wrc = new WarcReaderCompressed(new FileInputStream(gzipPath));
List<string> htmls = new ArrayList<String>();
```

Expert:
Job type: User-defined (Mailing list)
Causes: Large accumulated results (us, reproduced)
Fix suggestions: no

5. Bulk-load to HBase

User: I have to merge the byte[]s that have the same key. If merging is done with reduceByKey(), a lot of intermediate byte[] allocation and System.arraycopy() is executed, and it is too slow. So I had to resort to groupByKey(), and in the callback allocate the byte[] that has the total size of the byte[]s, and arraycopy() into it. groupByKey() works for this, since the size of the group is manageable in my application.
Expert: The problem is that you will **first collect and allocate many small byte[] in memory, and then merge them. If the total size of the byte[]s is very large, you run out of memory, as you observe.** If you want to do this, use more executor memory. You may find it's not worth the tradeoff of having more, smaller executors merging pieces of the overall byte[] array.

Job type: User-defined (Mailing list)
Causes: Large accumulated results (Expert)
Fix suggestions: no

6. Help alleviating OOM errors

User: our cluster seems to have a really hard time with OOM errors on the executor.
Expert: You need to configure the spark.shuffle.spill true again in the config, What is causing you to OOM, it could be that you are trying to just simply sortbykey & keys are bigger memory of executor causing the OOM, can you put the stack. When OOM occurs it could cause the RDD to spill to disk, the repeat task may be forced to read data from disk & cause the overall slowdown, not to mention the RDD may be send to different executor to be processed, are you seeing the slow tasks as process_local or node_local atleast?

Too few partitions: if one partition is too big, it may cause an OOM if there is not enough space to unroll the entire partition in memory. For the latter, I would reduce spark.storage.memoryFraction. Your application is using a lot of memory on its own: Spark be default assumes that it has 90% of the runtime memory in your JVM. **If your application is super memory-intensive (e.g. creates large data structures),** then I would either try to reduce the memory footprint of your application itself if possible, or reduce the amount of memory Spark thinks it owns.

Job type: User-defined (Mailing list)
Causes: Large accumulated results (Expert)
Fix suggestions: add partition number

7. Memory footprint of Calliope: Spark -> Cassandra writes

User: I'm generating N records of about 50 bytes each and using the UPDATE mutator to insert them into C*.   I get OOM if my memory is below 1GB per million of records, or about 50Mb of raw data (without counting any RDD/structural overhead). (See code [1]). I just tried the code you posted in the gist (https://gist.github.com/maasg/68de6016bffe5e71b78c) and it does give a OOM. It is cause of the data being generated locally and then paralellized -

Expert: You need to configure the spark.shuffle.spill true again in the config, What is causing you to OOM, it could be that you are trying to just simply sortbykey & keys are bigger memory of executor causing the OOM, can you put the stack. When OOM occurs it could cause the RDD to spill to disk, the repeat task may be forced to read data from disk & cause the overall slowdown, not to mention the RDD may be send to different executor to be processed, are you seeing the slow tasks as process_local or node_local atleast?

Job type: User-defined (Mailing list)
Causes: Large accumulated results (us, reproduced)
Fix suggestions: no

8. Folding an RDD in order

User: I have data that looks like this:

user_id, transaction_timestamp, transaction_amount

And I'm interested in doing a foldByKey on user_id to sum transaction amounts - taking care to note approximately when a user surpasses a total transaction threshold. I'm using RangePartitioner to make sure that data is ordered sequentially between partitions, and I'd also make sure that data is sorted within partitions, though I'm not sure how to do this exactly (I was going to look at the code for sortByKey to figure this out - I believe sorting in place in a mapPartitions should work).

Expert: you may use mutable Map for optimized performance. One thing to notice, foldByKey is a transformation, while aggregate is an action. The final result of the code above is a single Map object rather than an RDD. **If this map can be very large (say you have billions of users), then aggregate may OOM**.

Job type: User-defined (Mailing list)
Causes: Large accumulated results (Expert)
Fix suggestions: no

9. [GitHub] incubator-spark pull request: MLLIB-25: Implicit ALS runs out of m...

User: Completely correct, but there's a subtle but quite large memory problem here. **map() is going to create all of these matrices in memory at once, when they don't need to ever all exist at the same time.**
For example, if a partition has n = 100000 rows, and f = 200, then this intermediate product requires 32GB of heap. The computation will never work unless you can cough up workers with (more than) that much heap.

Expert:
Job type: User-defined (Mailing list)
Causes: Large accumulated results (User)
Fix suggestions: no

10. GroupByKey results in OOM - Any other alternative

User: There is a huge list of (key, value) pairs. I want to transform this to (key, distinct values) and then eventually to (key, distinct values count)

On small dataset

groupByKey().map(x => (x_1, x._2.distinct)) ...map(x => (x_1, x._2.distinct.count))

On large data set I am getting OOM.

Is there a way to represent Seq of values from groupByKey as RDD and then perform distinct over it ?
Expert:
Grouping by key is always problematic **since a key might have a huge number of values**. You can do a little better than grouping *all* values and *then* finding distinct values by using foldByKey, putting values into a Set. At least you end up with only distinct values in memory. (You don't need two maps either, right?) If the number of distinct values is still huge for some keys, consider the experimental method countApproxDistinctByKey
Job type: User-defined (Mailing list)
Causes: Large accumulated results + Hotspot Key (Expert, reproduced)
Fix suggestions: use (countApproxDistinctByKey)

9. Large data/results generated at driver

9.1 Spark Errors (9)

1. Q: Spark mllib svd gives: Java OutOfMemory Error

User: I am using the svd library of mllib to do some dimensionality reduction on a big matrix, the data is about 20G, and the spark memory is 60G, and I got the following warning and error message:
I think the error happens while java is copying an array. So I think this happens on the driver. How should I fix this? Use a larger driver memory?

The reason why I am getting the java memory error is because, **the computation for top eigenvectors are on the driver, so I need to make sure that I have enough memory on the driver node. When using `spark-submit` with `--driver-memory 5G`, the problem is solved.**

Expert:
The SVD operation you are calling happens on the driver. The covariance matrix is calculated on the cluster though. Where are you running out of memory? Driver right?

Job type: MLlib (StackOverflow)
Causes: Driver generate large results (Top eigenvectors) (User)
Fix suggestions: Add driver's memory space

2. A: How to use spark to generate huge amount of random integers?

User:
Notes: Now I just want to generate one number per line.

But it seems that when number of numbers gets larger, the program will report an error. Any idea with this piece of code?

Expert:
The current version is materializing the collection of random numbers in the memory of the driver. If that collection is very large, the driver will run out of memory.

Job type: User-defined (StackOverflow)
Causes: Driver generates large results (Expert)
Fix suggestions: no

3. Running out of memory Naive Bayes

User:
I've been trying to use the Naive Bayes classifier. Each example in the dataset is about 2 million features, only about 20-50 of which are non-zero, so the vectors are very sparse. I keep running out of memory though, even for about 1000 examples on 30gb RAM while the entire dataset is 4 million examples. And I would also like to note that I'm using the sparse vector class.
Expert:
Even the features are sparse, the conditional probabilities are stored in a dense matrix. With 200 labels and 2 million features, you need to store at least 4e8 doubles on the driver node. With multiple partitions, you may **need more memory on the driver**. Could you try reducing the number of partitions and giving driver more ram and see whether it can help
Job type: MLlib (Mailing list)
Causes: Driver generates large results (Expert, reproduced)
Fix suggestions: reduce the partition number

4. trouble with broadcast variables on pyspark

User:
I'm running into an issue when trying to broadcast large variables with pyspark.

A ~1GB array seems to be blowing up beyond the size of the driver machine's memory when it's pickled.

I've tried to get around this **by broadcasting smaller chunks of it one at a time. But I'm still running out of memory**, ostensibly because the intermediate pickled versions aren't getting garbage collected.
The driver JVM is hitting OutOfMemoryErrors, but the python process is taking even more memory.

Expert:
Job type: User-defined (Mailing list)
Causes: Driver generates/broadcasts large results (Expert)
Fix suggestions: no

5. advice on maintaining a production spark cluster?

User: I suspect that the loss of workers is tied to
jobs that run out of memory on the client side or our use of very large broadcast variables, but I don't have an isolated test case.
I'm open to general answers here: for example, perhaps we should simply be using mesos or yarn instead of stand-alone mode.

We've had occasional problems with running out of memory on the driver side (esp. **with large broadcast variables**) so that may be related.

Job type: User-defined (Mailing list)
Causes: Driver generates/broadcasts large results (User)
Fix suggestions: no

6. RowMatrix PCA out of heap space error

User: I got this error when trying to perform PCA on a sparse matrix, each row has a nominal length of 8000, and there are 36k rows. each row has on average 3 elements being non-zero.
I guess the total size is not that big.
Expert: The Gramian is 8000 x 8000, dense, and full of 8-byte doubles. It's symmetric so can get away with storing it in ~256MB. The catch is that it's going to send around copies of this 256MB array. You may easily be **running your driver out of memory** given all the overheads and copies, or your executors, of which there are probably 2 by default splitting 1GB.

Job type: MLlib (Mailing list)
Causes: Driver generates large results (User, reproduced)
Fix suggestions: no

7. newbie : java.lang.OutOfMemoryError: Java heap space

User: Disclaimer : Newbie (well, a returning user)
Setup :
20 nodes
-Dspark.executor.memory=40g , essentially tons of space for my usecase

Pretty straight forward join between two inputs

- 17G (distributed in 10 equally sized - 1.7g files)
- 49Mb (1 file)
I just need to join based on the keys and write out values from both as tuples
Expert: From the stack trace, it looks like the **driver program is dying trying to serialize data out to the workers**. My guess is that whatever machine you're running from has a relatively small default maximum heap size and trying to broadcast the 49MB file is causing it to run out of memory

Job type: User code (Mailing list)
Causes: Driver generates large results (Expert)
Fix suggestions: no

8. broadcast: OutOfMemoryError

User: i'm running into this OutOfMemory issue when i'm broadcasting a large array. what is the best way to handle this?

should i split the array into smaller arrays before broadcasting, and then combining them locally at each node?

Job type: User code (Mailing list)
Causes: Driver generates/broadcasts large results (User)
Fix suggestions: no

9. driver memory

User: In the application UI, it says my driver has 295 MB memory. I am trying to broadcast a variable that is 0.15 gigs and it is throwing OutOfMemory errors, so I am trying to see if by increasing the driver memory I can fix this. y trying to broadcast an array with 4 million elements, and a size of approximatively 150 MB. Every time I was trying to broadcast, I got an OutOfMemory error.

Job type: User code (Mailing list)
Causes: Driver generates/broadcasts large results (User)
Fix suggestions: no

10. Large results collected by driver

10.1 Spark Errors (16)

1. Q: Spark raises OutOfMemoryError
User: Take(all) used in the code
The file is about 20G and my computer have 8G ram, when I run the program in standalone mode, it raises the OutOfMemoryError:

Expert:
Spark can handle some case. But you are using `take` to f**orce Spark to fetch all of the data to an array(in memory)**. In such case, you should store them to files, like using `saveAsTextFile`.
If you are interested in looking at some of data, you can use `sample` or `takeSample`.

Job type: User-defined (StackOverflow)
Causes: Driver collect results (Expert)
Fix suggestions: no

2. Q: Spark OutOfMemoryError when adding executors

User:
Everything runs smoothly when I use few executors (10). But I got `OutOfMemoryError: Java heap space` on the driver when I try to use more executors (40). I think it might be related to the level of parallelism used (as indicated in https://spark.apache.org/docs/latest/tuning.html#level-of-parallelism).

Expert:
Do you mind testing 1.1-SNAPSHOT and allocating more memory to the driver? I think the problem is with the feature dimension. KDD data has more than 20M features and in v1.0.1, the driver collects the partial gradients one by one, sums them up, does the update, and then sends the new weights back to executors one by one. In 1.1-SNAPSHOT, we switched to multi-level tree aggregation and torrent broadcasting.

Job type: User-defined (StackOverflow)
Causes: Driver collects large results (User)
Fix suggestions: Multi-level tree aggregation (Expert)

3. Q: GraphX does not work with relatively big graphs

User:
Graph has 231359027 edges. And its file weights 4,524,716,369 bytes. Graph is represented in text format:

```
println(graph.edges.collect.length)     println(graph.vertices.collect.length)
```

Expert:

Job type: GraphX (StackOverflow)
Causes: Driver collects large results (User, reproduced)
Fix suggestions: Change collect().count() to count() (User)

4. Q: How to filter a RDD according to a function based another RDD in Spark?
User:
When the input data is very large, > 10GB for example, I always encounter a "java heap out of memory" error. I doubted if it's **caused by "weights.toArray.toMap", because it convert an distributed RDD to an Java object in JVM.** So I tried to filter with RDD directly:
Job type: User-defined (StackOverflow)
Causes: Driver collects large results (User)
Fix suggestions: no

5. A: How to iterate over large Cassandra table in small chunks in Spark
User:
In my test environment I have 1 Cassandra node and 3 Spark nodes. I want to iterate over apparently large table that has about 200k rows, each roughly taking 20-50KB.

I tried to only run `collect`, without `count` - in this case it just fails fast with `NoHostAvailableException`.

Expert:

Furthermore, you shouldn't use the collect action in your example because it will fetch all the rows in the driver application memory and may raise an out of memory exception. You can use the collect action only if you know for sure it will produce a small number of rows.

Job type: User-defined (StackOverflow)
Causes: Driver collects large results (Expert)
Fix suggestions: remove collect()

6. Memory allocation in the driver

User: It turns out that the second call to "b.first" will cause the spark driver to **allocate a VERY large piece of memory**. The first item of the first partition is very small (less than 200 bytes). However, the size of the entire first partition was about 380 MB. Apparently, my driver prepared itself to receive something large like an entire partition and allocated a chunk of appropriate size. Since I configured the driver to use only 256 MB, the allocation did not succeed and it ran out of memory.

Expert:
Job type: User-defined (Mailing list)
Causes: Driver collects large results (Expert)
Fix suggestions: no

7. something about rdd.collect

User: documents.flatMap(case words => words.map(w => (w, 1))).reduceByKey(_ + _).collect()
In driver's log, reduceByKey() is finished, but collect() seems always in run, just can't be finished.
In additional, there are about 200,000,000 words needs to be collected. Is it too large for collect()? But when words decreases to 1,000,000, it's okay!
Expert: collect essentially transfers all the data to the driver node. You definitely wouldn't want to collect 200 million words. It is a pretty large number and you can run out of memory on your driver with that much data.

Job type: User-defined (Mailing list)
Causes: Driver collects large results (User)
Fix suggestions: no

8. Driver OOM while using reduceByKey

User: I used 1g memory for the driver java process and got OOM error on driver side before reduceByKey. After analyzed the heap dump, the biggest object is org.apache.spark.MapStatus, which occupied over 900MB memory.
Expert: That hash map is just a list of where each task ran, it's not the actual data. How many map and reduce tasks do you have? Maybe you need to give the driver a bit more memory, or use fewer tasks (e.g. do reduceByKey(_ + _, 100) to use only 100 tasks).

Job type: User code (Mailing list)
Causes: Driver collects large results (Expert)
Fix suggestions: reduce task number

9. Beginner Question on driver memory issue (OOM).

User: Run the same action again. (OOM issue).

Observations

Running Linux top, **the driver process is definitely reaching the max memory usage**. My assumption was that calling collect on the RDD would only require high memory pressure while the collect function builds the result set and returns it. However it seems to be residing in memory as if its being cached or still hard-referenced somewhere.
Expert:

Job type: User code (Mailing list)
Causes: Driver collects large results (User)
Fix suggestions: no

10. Does foreach operation increase rdd lineage?

User: I'm writing a paralell mcmc program that having a very large dataset in memory, and need to update the dataset in-memory and avoid creating additional copy. Should I choose a foreach operation on rdd to express the change? or I have to create a new rdd after each sampling process?
Expert: Do you mean "Gibbs sampling" ? Actually, foreach is an action, it will collect all data from workers to driver. You will get OOM complained by JVM.

Job type: User code (Mailing list)
Causes: Driver collects large results (Expert)
Fix suggestions: no

11. RDD with a Map

User: for this to actually materialize I do collect

val groupedAndCollected=groupedWithValues.collect()

I get an Array[String,List[String]].

I am trying to figure out if there is a way for me to get Map[String,List[String]] (a multimap), or to create an RDD[Map[String,List[String]]]
Expert: At last, be careful if you are processing large volume of data, since groupByKey is an expensive transformation, and collecting all the data to driver side may simply cause OOM if the data can't fit in the driver node.

Job type: User code (Mailing list)
Causes: Driver collects large results (Expert)
Fix suggestions: no

12. How to efficiently join this two complicated rdds

User: For each line in RDD one, we need to use the keys of the line to search the value according to the key in RDD of type two. And finally get the sum of these values. we have implement this way, we use pyspark, and standalone mode**.** **We collect the new RDD2 in each iteration**. The java heap memory costed by the driver program increases Gradually. And finally Collapse with OutOfMemory Error.

We have done some tests, in each iteration, we simply collect a vector. This Little Simple problem also costed more and more java heap memory, and finally raised OutOfMemory.

Expert: We have done some experiment using data which has much small size but same form. The method above will cost more than 10 mins while using collectAsMap function to collect RDD2 and sending it to each worker will cost 2 mins. But the second method will get outOfMemery error while we try the big data.
Job type: User code (Mailing list)
Causes: Driver collects large results (User)
Fix suggestions: no

13. java.lang.OutOfMemoryError while running SVD MLLib example

User: I am new to Spark. I have downloaded Spark 1.1.0 and trying to run the TallSkinnySVD.scala example with different input data sizes. I tried with input data with 1000X1000 matrix, 5000X5000 matrix.
Though I had faced some Java Heap issues I added following parameters in "spark-defaults.conf"

```
    spark.driver.memory          5g
    spark.executor.memory    6g
```

Now, I am trying with 7000X7000 input matrix, but it fails with OutofMemory error.
I tried by setting executor memoryto 8g but didn't worked.
I also tried by setting persist to MEMORY_AND_DISK level but no luck.

```
    rows.persist(StorageLevel.MEMORY_AND_DISK)
```

Expert: 7000x7000 is not tall-and-skinny matrix. Storing the dense matrix requires 784MB. The driver needs more storage for collecting result from executors as well as making a copy for LAPACK's dgesvd. So you need more memory. Do you need the full SVD? If not, try to use a small k

Job type: MLlib (Mailing list)
Causes: Driver collects large results (User, reproduced)
Fix suggestions: use a small k (parameter)

14. Maximum size of vector that reduce can handle

User: I am trying to measure the Spark reduce performance for big vectors. My motivation is related to machine learning gradient. Gradient is a vector that is computed on each worker and then all results need to be summed up and broadcasted back to workers. For example, present machine learning applications involve very long parameter vectors, for deep neural networks it can be up to 2Billions.
"spark.driver.maxResultSize 0" needs to set in order to run this code. I also needed to change "java.io.tmpdir" and "spark.local.dir" folders because my /tmp folder which is default, was too small and Spark swaps heavily into this folder. Without these settings I get either "no space left on device" or "out of memory" exceptions.
Please try treeReduce instead which is what we do in linear regression and logistic regression.

Expert: When you use `reduce` to aggregate the vectors, those will actually be pulled into driver, and merged over there. Obviously, it's not scaleable given you are doing deep neural networks which have so many coefficients. For `reduce`, it's an action that will collect all the data from mapper to driver, and perform the aggregation in driver. As a result, if the output from the mapper is very large, and the numbers of partitions in mapper are large, it might cause a problem.

For `treeReduce`, as the name indicates, the way it works is in the first layer, it aggregates the output of the mappers two by two resulting half of the numbers of output. And then, we continuously do the aggregation layer by layer. The final aggregation will be done in driver but in this time, the numbers of data are small.

60m-vector costs 480MB memory. You have 12 of them to be reduced to the driver. So you need ~6GB memory not counting the temp vectors generated from '_+_'. You need to increase driver memory to make it work. That being said, ~10^7 hits the limit for the current impl of glm.

Job type: MLlib (Mailing list)
Causes: Driver collects large results (User)
Fix suggestions: tree reduce

15. take() reads every partition if the first one is empty

User: Wouldn't a better implementation strategy be

numPartsToTry = partsScanned * 2

instead of numPartsToTry = totalParts - 1
the logic for take() reads ALL partitions if the first one (or first k) are empty. **This has actually lead to OOMs when we had many partitions (thousands) and unfortunately the first one was empty**.

Expert:
Job type: User-defined (Mailing list)
Causes: Driver collects large results (User)
Fix suggestions: no

16. Q: Memory efficient way of union a sequence of RDDs from Files in Apache Spark
User:
I often run into out of memory situations even on 100 GB plus Machines. I run Spark in the application itself. I tried to tweak a little bit, but I am not able to perform this operation on more than 10 GB of textual data. The clear bottleneck of my implementation is the union of the previously computed RDDs, that where the out of memory exception comes from.

Expert:
for my use case that issue made the word2vec spark implementation a bit useless. Thus I used spark for massaging my corpus but not for actually getting the vectors.

- As other suggested stay away from calling `rdd.union`.
- Also I think `.toList` will probably gather every line from the RDD and collect it in your Driver Machine (the one used to submit the task) probably this is why you are getting out-of-memory. You should totally avoid turning the RDD into a list!

Job type: User-defined (StackOverflow)
Causes: Large results collected by driver (Expert)
Fix suggestions: avoid turning the RDD into a list

11. Errors that their root causes are unknown (153)

11.1 Hadoop Errors (95)

1. Q: OutofMemoryError when reading a local file via DistributedCache
User: However, when I execute it in the Hadoop cluster (fully distributed mode), I get an "OutOfMemoryError: Java heap space"

Expert:

Job type: User-defined (StackOverflow)
Causes: Unknown

2. Q: Hadoop searching words from one file in another file
User: I want to build a hadoop application which can read words from one file and search in another file.
If the word exists - it has to write to one output file If the word doesn't exist - it has to write to another output file
I tried a few examples in hadoop. I have two questions
Two files are approximately 200MB each. Checking every word in another file might cause out of memory. Is there an alternative way of doing this?
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

3. Q: Java Heap space error in Hadoop
User: I am currently reading file of 50 MB in reducer setup step successfully, But when file is larger than that approx(500MB) it gives me "out of memory error".
Expert: Increase the heap size

Job type: User-defined (StackOverflow)
Causes: Unknown (maybe large external data)

4. Q: Java heap space error while running hadoop
User: I am trying to run hadoop-examples.jar-1.2.1 from hadoop examples. I am using 64 -bit Linux system. No i have not tried that. The memory i set up for the virtual machine is IGB. Each time I run the job it shows Java heap space exception and the job fails.
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

5. Q: Out of heap error when creating Index in Apache Hive
User: when we execute the “alter” statement to actually build it this fails with "java.lang.OutOfMemoryError: Java heap space". For a partitioned table the index for each partition seems to be built individually so in this case about 1500 separate jobs are launched (we have tried both to let them run one at a time and several in parallel with the same result) and a large number of jobs actually seem to run as they should but after a while we start to see jobs failing and after this point most of the remaining jobs actually fail.
Expert:
Job type: Apache Hive (StackOverflow)
Causes: Unknown

6. Q: Error: Java heap space
User: running the hadoop example :
$bin/hadoop jar hadoop-examples-1.0.4.jar grep input output 'dfs[a-z.]+'
In log, I am getting the error.
Expert: add the heap size
Job type: User-defined (StackOverflow)
Causes: Unknown

7. [Q: increase jvm heap space while runnig from hadoop unix]

User: I am running a java class test.java from hadoop command :

$ hadoop test

I am using a stringBuilder, and its size is going out of memory :

Expert:

Causes: Unknown

8. [Q: PIG using HCatLoader, Java heap space error]

User: When I try to load a table created in hive into pig using HCatloader, it is giving the Java heap space error.The detailes are as follow

Expert:

Job type: Apache Pig (StackOverflow)

Causes: Unknown

9. [Q: Error running child : java.lang.OutOfMemoryError: Java heap space]

User: I have read a lot on the internet, but found no solution for my problem. I use Hadoop 2.6.0.

The main goal for the *MapReduce* is to run through a *SequenceFile* and do some analysis on the key/value pairs.

Expert:

Job type: User-defined (StackOverflow)

Causes: Unknown

10. [Q: Hadoop: Can you silently discard a failed map task?]

User: I am processing large amounts of data using hadoop MapReduce. The problem is that, ocassionaly, a corrupt file causes Map task to throw a java heap space error or something similar.

Expert:

Job type: User-defined (StackOverflow)

Causes: Unknown

11. [Q: Querstion regarding hadoop-env.sh]

User: I am Facing Error: Java heap space and Error: GC overhead limit exceeded

So i started looking into hadoop-env.sh.

so thats what i understand so far, Please correct me if i am wrong.

Expert:

Job type: User-defined (StackOverflow)

Causes: Unknown

12. [Q: how to change mapper memory requirement in hadoop?]

User: In a map-reduce job, i got the error "java.lang.OutOfMemoryError: Java heap space". Since I get this error in a mapper function; I thought that when I lower the input size to the mapper I will have no more error, so I changed the mapred.max.split.size to a much more lower value.

Then, I started the job again and i saw that "number of mapper tasks to be executed" has increased, so i thought that lowering mapred.max.split.size was a good idea: more mappers with lower memory requirements.

BUT, I got the "java.lang.OutOfMemoryError: Java heap space" error again, again and again.

Job type: User-defined (StackOverflow)

Causes: Unknown

13. [Q: Hadoop conf to determine num map tasks]

User: I have a job, like all my Hadoop jobs, it seems to have a total of 2 map tasks when running from what I can see in the Hadoop interface. However, this means it is loading so much data that I get a Java Heap Space error.

I've tried setting many different conf properties in my Hadoop cluster to make the job split into more tasks but nothing seems to have any effect.
I have tried setting mapreduce.input.fileinputformat.split.maxsize, mapred.max.split.size, dfs.block.size but none seem to have any effect.
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

14. A: Java Heap Space Error in running mahout item similarity job on Amazon EMR
User: I am trying to run a mahout item similarity job on a input consists of ~250 Million Pairs(row) in a Amazon EMR Cluster(m3.2xLarge,10 core nodes).I am facing Java Heap Size error while running the similarity job.
Expert:
Job type: Apache Mahout (StackOverflow)
Causes: Unknown

15. Q: Mahout on EMR Error: Java heap space
User: I've ran a clustering job on EMR. The dataset is huge. Everything worked well until:
Expert:
Job type: Apache Mahout (StackOverflow)
Causes: Unknown

16. Q: How to handle unsplittable 500 MB+ input files in hadoop?
User: However, I noticed that hadoop works very poorly with output values that can sometimes be really big (700 MB is the biggest I've seen). In various places in the MapReduce framework, **entire files are stored in memory, sometimes twice or even three times.** I frequently encounter out of memory errors, even with a java heap size of 6 GB.
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

17. Q: Hive Issue - java.lang.OutOfMemoryError: Java heap space
User: I have started hive server and i am trying to login to Hive and run a basic command. Below is snapshot of the issue.
Expert:
Job type: Apache Hive (StackOverflow)
Causes: Unknown

18. A: hive collect_set crashes query
User: I've got the following table:
where I remove the collect_set command. So my question: Has anybody an idea why collect_set might fail in this case?
Expert: This is probably the memory problem, since collect_set aggregates data in the memory.
Job type: Apache Hive (StackOverflow)
Causes: Unknown

19. Q: MapReduce jobs in hive-0.8.1-cdh4.0.1 Failed.
User: Queries in **hive-0.8.1-cdh4.0.1** that invoke the Reducer results in Task Failed. The queries having MAPJOIn is working fine but JOIN gives error.
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

20. Q: how to determine the size of “mapred.child.java.opts” and HADOOP_CLIENT_OPTS in mahout canopy

User: I’ve got a "dictionary.file-0" file and its size is about 50M.
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

21. Q: setting hadoop job configuration programmatically

User: I am getting OOM exception (Java heap space) for reduce child. I read in the documentation that increasing the value of mapred.reduce.child.java.opts to -Xmx512M or more would help. Since I am not the admin, I cannot change that value in mapred-site.xml. I would like to set that value only for my job through the java program. I tried setting it using Configuration class as follows, but that didn't work.
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

22. Q: Is it possible to intervene if a task fails?

User: I have a mapreduce job running on many urls and parsing them. I need way to handle a scenario in which one parsing task crashes on a fatal error like OOM error. In the normal hadoop behaivour a task is retried for a defined number of time and than the job fails. the problem is with urls that are corrupted in some way causing this error. These urls will fail in all the retries.
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

23. Q: Limit CPU / Stack for Java method call?

User: I am using an NLP library (Stanford NER) that throws OOM errors for rare input documents.
I plan to eventually isolate these documents and figure out what about them causes the errors, but this is hard to do (I'm running in Hadoop, so I just know the error occurs 17% through split 379/500 or something like that). As an interim solution, I'd like to be able to apply a CPU and memory limit to this particular call.
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

24. Q: Hadoop YARN Map Task running out of physical and virtual memory

User: have the following method that I run from my map task in a multithreaded execution , however this works fine in a standalone mod e, but when I runt this in Hadoop YARN it runs out of the physical memory of 1GB and the virtual memory also shoots up.
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

25. Q: Why the identity mapper can get out of memory?

User: After some hours of researching and trial and error I realized that the machines I provisioned for the TASK group were small instances with not much memory and, more interestingly, that the point in which I was running out of memory was during shuffling instead of mapping.
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

25. Q: Java Heap space error in Hadoop
User: My problem is with hadoop heap memory. I am currently reading file of 50 MB in reducer setup step successfully, But when file is larger than that approx(500MB) it gives me "out of memory error". When i browse through http:/ip_address:50070/dfshealth.html#tab-overview it gives me below information.
Expert: The reason why you use something like hadoop is because you cant fit the entire data set into memory. Either you don't change the logic and try to find a computer that's big enough or you parallelize the algorithm and exploit hadoop.
Job type: User-defined (StackOverflow)
Causes: Unknown (Maybe large external data)

26. Q: Will reducer out of java heap space
User: My question is how to deal with java out of space problem, I added some property configuration into xml file, but it didn't work. Increasing number of reducers doesn't work for me either. Because in my program every reducer needs large sparse whole matrix, and I am not allowed to change this logic. Yet every reducer will receive an entry with column id as key, and column vector as value.Is there any way I can get out of this dilemma?
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

27. Q: Hadoop: Heap space and gc problems
User: My algorithm works fine for small datasets, but for a medium dataset has heap space problems. My algorithm reaches a certain tree level and then it goes out of heap space, or has gc overhead problems. At that point, i made some calculations and i saw that every task doesnt need more than 100MB memory. So for 8 tasks, i am using about 800MB of memory. I don't know what is going on. I even updated my hadoop-env.sh file with these lines:
Expert: add heap size
Job type: User-defined (StackOverflow)
Causes: unknown

28. A: Hadoop JobClient: Error Reading task output
User: I had a similar problem and was able to find a solution. The problem lies on how hadoop deals with smaller files. In my case, I had about 150 text files that added up to 10MB. Because of how the files are "divided" into blocks the system runs out of memory pretty quickly.
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

29. A: Pig: Hadoop jobs Fail
User: I have a pig script that queries data from a csv file.
The script has been tested locally with small and large .csv files.
In Small Cluster: It starts with processing the scripts, and fails after completing 40% of the call
The error is just, Failed to read data from "path to file"
What I infer is that, The script could read the file, but there is some connection drop, a message lose
But I get the above mentioned error only.
Expert:
Job type: Apache Pig (StackOverflow)
Causes: Unknown

30. Subject: New type of JOIN specialized for filtering
User: In many cases I am close to be able to use a replicated join (100-150 MB of data) but it still blows up despite upping the Java heap to a few GB. e.g.

```
A = LOAD 'bigdata'
B = LOAD 'smalldata'
C = FOREACH B GENERETE key1, key2      -- < 150 MB
D = JOIN A BY (key1, key2), C BY (key1, key2) USING 'replicated'
....
java.lang.OutOfMemoryError: Java heap space
```

I looked at the code of the POFRJoin and its primary goal is to JOIN some extra columns. Would it make sense to have a new type of join for efficiently doing some filtering on the map side? (something similar to LookupInFiles but working transparently with Pig relations) Do better techniques exist already?
Expert: Seems like it would be more memory-efficient to pull the group keys out of the hashmap value:
e.g.:
x: key, a, b
y: key, c, d
join x by key, y by key using 'replicated';

Current FRJoin:
construct a hashmap of { key --> (key, a, b) }
Proposed optimization:
construct a hashmap of { key --> (a, b) }, keep track of where key should
be inserted to reconstruct the tuple.

That *almost* gets us where you suggest -- a pure filtering join would let us drop the whole hashmap and replace with a hashset -- but it also lets us avoid some extra complexity so that may be a good thing. I think giving users too many types of joins may be a bad thing.

If we implemented the IN operator, we could do what you suggest without the complexity overhead.. IN could take a relation or bag of single-valued tuples, or a list of scalars, and for all 3 cases it would build up an in-memory hashset.

Job type: Apache Pig (Mailing list)
Causes: Unknown

31. Q: How to run large Mahout fuzzy kmeans clustering without running out of memory?

User: I am running Mahout 0.7 fuzzy k-means clustering on Amazon's EMR (AMI 2.3.1) and I am running out of memory.
Expert: So 4 * (100000 entries) * (20 bytes/entry) * (128 clusters) = 1.024G. This algorithm is a memory hog.
Job type: Apache Mahout (StackOverflow)
Causes: Unknown

32. Q: Memory problems with Java in the context of Hadoop

User: These errors occur when I hash the input records and only then for the moment. During the hashing I have quite many for loops which, produce a lot of temporary objects. For this reason I get the 1) problem. So, I solved the 1) problem by setting K = StringBuilder which is a final class. In other words I reduced the amount of temporary objects by having only few objects which their value, content changes but not themselves.
Expert: Use an ArrayList instead of a LinkedList and it will use a lot less memory. Also I suggest using a HashMap instead of Hastable as the later is a legacy class.
Job type: User-defined (StackOverflow)
Causes: Unknown

33. Subject: 0.9.1 out of memory problem

User: I'm having an out of memory problem that seems rather weird to me. Perhaps you can help me.

bySite = GROUP wset BY site; report = FOREACH bySite {   duids = DISTINCT wset.duid;   GENERATE group, COUNT(duids), SUM(wset.replicas), SUM(wset.nbfiles), SUM(wset.rnbfiles), SUM(wset.length), SUM(wset.rlength); }; STORE report INTO 'testfile.out';
If I omit the COUNT(DISTINCT), it works brilliantly and fast. With the COUNT(DISTINCT) it dies like this. Now, I don't know where to go from here. I'm running Hadoop and Pig with default settings, except I've increased child.opts to -Xmx1024M (24GB machines) so it would be great if you could tell me what to do, because I'm stuck.
Expert:
Job type: Apache Pig (Mailing list)
Causes: Unknown

34. A: Hadoop example job fails in Standalone mode with: "Unable to load native-hadoop library"
User: I'm trying to get the simplest Hadoop "hello world" setup to work, but when I run the following command:
hadoop jar /usr/share/hadoop/hadoop-examples-1.0.4.jar grep input output 'dfs[a-z.]+'
Expert:
Job type: User-defined (StackOverflow)
Causes: Unknown

35. OOM Error Map output copy.

User: I am encountering the following out-of-memory error during the reduce phase of a large job. I am currently using 12 reducers and I can't increase this count by much to ensure availability of reduce slots for other users. mapred.job.shuffle.input.buffer.percent --> 0.70
Fortunately, our requirements for this job changed, allowing me to use a combiner that ended up reducing the data that was being pumped to the reducers and the problem went away.
Expert:
Job type: User-defined (Mailing list)
Causes: Unknown

36. java.lang.OutOfMemoryError: Direct buffer memory

Job type: User-defined (Mailing list)
Causes: Unknown

36. OOME only with large datasets

Job type: User-defined (Mailing list)
Causes: Unknown

37. reducer outofmemoryerror

Job type: User-defined (Mailing list)
Causes: Unknown

38. Nor "OOM Java Heap Space" neither "GC OverHead Limit Exeeceded"

Job type: User-defined (Mailing list)
Causes: Unknown

39. Yarn container out of memory when using large memory mapped file

Job type: User-defined (Mailing list)
Causes: Unknown

40. ReducerTask OOM failure

Job type: User-defined (Mailing list)
Causes: Unknown

41. Reducer Out of Memory
Job type: User-defined (Mailing list)
Causes: Unknown

42. out of memory error

Job type: User-defined (Mailing list)
Causes: Unknown

43. RE: out of memory running examples

Job type: User-defined (Mailing list)
Causes: Unknown

44. Reduce tasks running out of memory on small hadoop cluster

Job type: User-defined (Mailing list)

45. Map Task is failing with out of memory issue

Job type: User-defined (Mailing list)

46. Out of Memory error in reduce shuffling phase when compression is turned on

Job type: User-defined (Mailing list)

47. java.lang.OutOfMemoryError: Direct buffer memory

Job type: User-defined (Mailing list)

48. OOM error with large # of map tasks

Job type: User-defined (Mailing list)

49. OutOfMemory error processing large amounts of gz files

Job type: User-defined (Mailing list)

50. OutofMemory Error, inspite of large amounts provided

Job type: User-defined (Mailing list)

51. OutOfMemoryError with map jobs

Job type: User-defined (Mailing list)

52. Caused by: java.lang.OutOfMemoryError: Java heap space - Copy Phase

Job type: User-defined (Mailing list)

53. OOME only with large datasets

Job type: User-defined (Mailing list)

54. Mapper OutOfMemoryError Revisited !!

Job type: User-defined (Mailing list)

55. OutOfMemory Error

Job type: User-defined (Mailing list)

56. reducer outofmemoryerror

Job type: User-defined (Mailing list)

57. reducer out of memory?

Job type: User-defined (Mailing list)

58. out of memory running examples

Job type: User-defined (Mailing list)

59. Out of memory in identity mapper?

Job type: User-defined (Mailing list)

60. Canopy generation out of memory troubleshooting

Job type: User-defined (Mailing list)

61. out of memory for Reducer possible?

Job type: Pig (Mailing list)

62. Java heap size increase caused MORE out of memory exceptions.

Job type: User-defined (Mailing list)

63. Yarn container out of memory when using large memory mapped file

Job type: User-defined (Mailing list)

64. Has anyone else seen out of memory errors at the start of combiner tasks?

Job type: User-defined (Mailing list)

65. Hadoop 1.2.1 corrupt after restart from out of heap memory exception

Job type: User-defined (Mailing list)

66. Out of Java heap space

Job type: User-defined (Mailing list)

67. Out of heap space errors on TTs

Job type: User-defined (Mailing list)

68. OutOfMemoryError of PIG job (UDF loads big file)

Job type: Pig (Mailing list)

69. OutOfMemoryError: Cannot create GC thread. Out of system resources

Job type: User-defined (Mailing list)

70. Shuffle In Memory OutOfMemoryError

Job type: User-defined (Mailing list)

71. Hashing two relations

Job type: User-defined (Mailing list)

72. OOM Error Map output copy.

Job type: User-defined (Mailing list)

73. OutOfMemoryError during reduce shuffle

Job type: User-defined (Mailing list)

74. Caused by: java.lang.OutOfMemoryError: Java heap space - Copy Phase

Job type: User-defined (Mailing list)

75. java.lang.OutOfMemoryError: Java heap space

Job type: Pig (Mailing list)

76. OutOfMemory during Plain Java MapReduce

Job type: User-defined (Mailing list)

77. OutOfMemoryError: unable to create new native thread

Job type: User-defined (Mailing list)

78. Reduce java.lang.OutOfMemoryError

Job type: User-defined (Mailing list)

79. OutOfMemory in ReduceTaskReduceCopierMapOutputCopier.shuffleInMemory

Job type: User-defined (Mailing list)

80. ReducerTask OOM failure

Job type: User-defined (Mailing list)

81. OOM error and then system hangs

Job type: User-defined (Mailing list)

82. how to prevent JAVA HEAP OOM happen in shuffle process in a MR job?

Job type: User-defined (Mailing list)

83. java.lang.OutOfMemoryError: Java heap space

Job type: User-defined (Mailing list)

84. java.lang.OutOfMemoryError: GC overhead limit exceeded

Job type: User-defined (Mailing list)

85. memoryjava.lang.OutOfMemoryError related with number of reducer?

Job type: User-defined (Mailing list)

86. Possible memory "leak" in MapTask$MapOutputBuffer

Job type: User-defined (Mailing list)

87. about the exception in mapreduce program?

Job type: User-defined (Mailing list)

88. MapReduce failure

Job type: User-defined (Mailing list)

89. High memory usage in Reducer

Job type: User-defined (Mailing list)

90. Child JVM memory allocation / Usage

Job type: User-defined (Mailing list)

91. Memory exception in the mapper

Job type: User-defined (Mailing list)

92. Killed : GC overhead limit exceeded

Job type: User-defined (Mailing list)

93. Solving "heap size error"

Job type: Apache Mahout (Mailing list)

94. heap size problem durning mapreduce

Job type: User-defined (Mailing list)

95. Error with Heap Space.

Job type: User-defined (Mailing list)

11.2 Spark Errors (58)

1. Q: OOM in spark pagerank

User:
When running graphX Page rank algorithm for 60 GB wiki data, the following error occurs. Please help.

The driver memory is 256m and executor memory is 6g. I tried increasing the driver memory as I am sorting the result and displaying first 100 pages.

Expert:
Pattern: Unknown (Broadcast related)

2. Q: Spark groupBy OutOfMemory woes

User:
I'm doing a simple groupBy on a fairly small dataset (80 files in HDFS, few gigs in total). I'm running Spark on 8 low-memory machines in a yarn cluster, i.e. something along the lines of:

Expert:
http://apache-spark-user-list.1001560.n3.nabble.com/Understanding-RDD-GroupBy-OutOfMemory-Exceptions-td11427.html#a11487

Patrick Wendell shed some light on the details of the groupBy operator on the mailing list. The takeaway message is the following:

Within a partition things will spill [...] This spilling can only occur *across keys* at the moment. Spilling cannot occur within a key at present. [...] Spilling within one key for GroupBy's is likely to end up in the next release of Spark, Spark 1.2. [...] If the goal is literally to just write out to disk all the values associated with each group, and the values associated with a single group are larger than fit in memory, this cannot be accomplished right now with the groupBy operator.

He further suggests a work-around:

The best way to work around this depends a bit on what you are trying to do with the data down stream. Typically approaches involve sub-dividing any very large groups, for instance, appending a hashed value in a small range (1-10) to large keys. Then your downstream code has to deal with aggregating partial values for each group. If your goal is just to lay each group out sequentially on disk on one big file, you can call `sortByKey` with a hashed suffix as well. The sort functions are externalized in Spark 1.1 (which is in pre-release).

Pattern: Large group, large data partitions

3. Q: Configure Java heap space with Spark

I'm trying to create a file with few hundred mega bytes by oversampling a small array in spark and save as object file to hdfs system created by spark-ec2 script:

```
//Oversampling repNum LabeledPoints from the array above val overSample = labelPts.takeSample(true, repNum, 1)
```

Then it throws a `EXCEPTION: java.lang.OutOfMemoryError: Java heap space`. I don't know what's wrong with it because if my `repNum` is set to 6000000, there will be no error and the output file is around 490m, so I suspect that the java heap space is still capped by 512m, however the I've set `--executor-memory=4g` and the worknode in this cluster has 7.5GB memory. What's the problem here?

Pattern: **Unknown**

4. Q: Calculate eccentricity of 5 vertices : Java heap space exeption

User :

have directed graph. Text file contains 5 million edges in format: sourceVertexId targetVertexId

and it size is approximately 54 Gb.

I want to load this graph using GraphLoader.edgeListFile and then using algorithm from this tutorial: https://spark.apache.org/docs/latest/graphx-programming-guide.html#pregel-api, I want to find eccentricity of 5 vertices.

Expert: None

Pattern: Unknown

5. Q: Error when trying to run algorithm in Spark

User:

```
hafidz@localhost dga]$ /opt/dga/dga-mr1-graphx pr -i sna_exp_comma.csv -o pr_sna.txt -n testPageRank -m spark://localhost.localdomain:7077 --S spark.executor.memory=1g  --S spark.worker.timeout=400  --S spark.driver.memory=1g
```

Expert:
Pattern: Unknown

6. Q: spark mllib memory error on svd (single machine)
User:
When I try to compute the principal components I get a memory error:

Expert:

Pattern: Unknown

7. Q: Spark throwing Out of Memory error
User:
When I try to load and process all data in the Cassandra table using spark context object, I'm getting an error during processing. So, I'm trying to use a looping mechanism to read chunks of data at a time from one table, process them and put them in another table.

Expert:
You are running a query inside the for loop. If the 'value' column is not a key/indexed column, Spark will load the table into memory and then filter on the value. This will certainly cause an OOM.

Pattern: Unknown

8. Q: Spark PCA OutOfMemory error on small number of columns and rows
User:
I am attempting to perform Spark MLLib PCA (using Scala) on a RowMatrix with 2168 columns, and a large number of rows. However, I have observed that even with as few as 2 rows in the matrix (a 112KB text file), the following error is always produced, at the same job step:

Pattern: Unknown

9. Q: Spark OutOfMemory error on small text data
User:
I am working on implementing an algorithm and testing it on medium-sized data in Spark (the Scala interface) on a local node. I am starting with very simple processing and I'm getting `java.lang.OutOfMemoryError: Java heap space` even though I'm pretty sure the data isn't big enough for such an error to be reasonable. Here is the minimal breaking code:

Expert:
So, thanks to all those small strings, your data in memory is roughly 5x the size 'at rest'. Still, 200k lines of that data makes up for roughly 500MB. This might indicate that your executor is operating at the default valie of 512MB. Try setting 'spark.executor.memory' to a higher value, but also consider a heap size >8Gb to confortably work with Spark.

Pattern: Unknown

10. Q: Out of memory exception during TFIDF generation for use in Spark's MLlib

User:
https://chimpler.wordpress.com/2014/06/11/classifiying-documents-using-naive-bayes-on-apache-spark-mllib/
Memory overflow and GC issues occur while collecting idfs for all the terms. To give an idea of scale, I am reading around 615,000(around 4GB of text data) small sized documents from HBase and running the spark program with 8 cores and 6GB of executor memory. I have tried increasing the parallelism level and shuffle memory fraction but to no avail.
Expert:
Pattern: Unknown

11. Q: OutOfMemoryError while Logistic regression in SparkR

User:
I have successfully installed Apache Spark, Hadoop over Ubuntu 12.04 (Single standalone mode) for Logistic regression. Also tested with small csv dataset but it doesnt work over large dataset having 269369 rows.

Expert:
As a back-of-the-envelope calculation, assuming each entry in your dataset takes 4 bytes, the whole file in memory would cost `269369 * 541 * 4 bytes ~= 560MB`
Pattern: Unknown

12. Q: I am getting the executor running beyond memory limits when running big join in spark

User:
I am getting the following error in the driver of a big join on spark.

We have 3 nodes with 32GB of ram and total input size of join is 150GB. (The same app is running properly when input file size is 50GB)

I have set storage.memoryFraction to 0.2 and shuffle.memoryFraction to 0.2. But still keep on getting the running beyong physical limits error

Pattern: Big join (Unknown)

13. Running out of memory Naive Bayes

14. Kafka streaming out of memory

15. Serializer or Out-of-Memory issues?

16. OUT OF MEMORY ERROR: HEAP SPACE

17. Out of Memory - Spark Job server

18. Driver fail with out of memory exception

19. Running out of memory on livejournal

20.MLlib Logistic Regression run out of memory

21.Executor out of memory and gets killed

22. out of memory errors -- per core memory limits?

23.help me: Out of memory when spark streaming

24.Out of memory exception in MLlib's naive baye's classification training

25.Meaning of persistence levels -- setting persistence causing out of memory errors with pyspark

26.Spark SQL 1.1.0 - large insert into parquet runs out of memory

27.OutOfMemory of spark streaming on standalone in 5 minutes, but run normal on local mode

28.JVM Memory Woes

29.Pyspark Memory Woes

30.OOM with groupBy + saveAsTextFile

31.trouble with "join" on large RDDs

32.Spark app throwing java.lang.OutOfMemoryError: GC overhead limit exceeded

33.java.lang.OutOfMemoryError: Requested array size exceeds VM limit

34.OutofMemoryError when generating output

35.Getting java.lang.OutOfMemoryError when calling mkString on a parition

36. newbie : java.lang.OutOfMemoryError: Java heap space

37.[Streaming]Executor OOM

38.OOM for HiveFromSpark example

39.OOM writing out sorted RDD

40.spark master OOME from maxMbInFlight buffers

41.Beginner Question on driver memory issue (OOM).

42.Spark-Shell: OOM: GC overhead limit exceeded

43.OOM Java heap space error on saveAsTextFile

44.pyspark sc.parallelize running OOM with smallish data

45.[PySpark] large # of partitions causes OOM

46.OOM - Requested array size exceeds VM limit

47.OOM when calling cache on RDD with big data

48.Running Wordcount on large file stucks and throws OOM exception

49.What the initial steps to understand root of this OOM exception

50.OutOfMemory Error

51.Cannot get rid of OutOfMemory

52.OutOfMemory error in Spark Core

53. OutOfMemory : Java heap space error

54.OutofMemory: Failed on spark/examples/bagel/WikipediaPageRank.scala

55.mllib svd gives: Java OutOfMemory Error

56.Scala PCA OutOfMemory error on small number of columns

57.OOM when making bins in BinaryClassificationMetrics ?

58. Re: Out of memory on large RDDs