# An Empirical Study of the Out of Memory Errors in Data-parallel Applications

Hadoop + Spark:

Caused Identified by:

|  |  |  |  |
| --- | --- | --- | --- |
| **Cause identified by** | **Hadoop** | **Spark** | **Total** |
| Users themselves | 12 | 33 | 45 |
| Experts | 37 | 29 | 66 |
| Us | 7 | 5 | 12 |
| Total | 56 | 67 | 123 |

Cases:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sources | User-defined code | Pig | Hive | Mahout | Cloud9 | GraphX | MLlib | Total (Reproduced) |
| StackOverflow | 20 | 4 | 2 | 4 |  |  |  | 30 (16) |
| Hadoop mailing list | 5 | 5 | 1 |  | 1 |  |  | 12 (6) |
| Developer’s blogs | 2 | 1 |  |  |  |  |  | 3 (2) |
| MapReduce books | 8 | 3 |  |  |  |  |  | 11 (2) |
| StackOverflow | 16 |  |  |  |  | 1 | 2 | 19 (3) |
| Spark mailing list | 42 |  |  |  |  | 1 | 5 | 48 (14) |
| Total | 93 | 13 | 3 | 4 | 1 | 2 | 7 | 123 (43) |

Cause patterns:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Category | Pattern | Pattern description | Hadoop | Spark | Total |
| Large data stored in memory | Large buffered data |  | 6 | 2 | 8 |
| Large cached data |  | 0 | 7 | 7 |
| Abnormal dataflow | Improper data partition |  | 3 | 14 | 17 |
| Hotspot key |  | 2+13 | 7 + 1 | 9+14 |
| Large single record |  | 3+3 | 1 | 4+3 |
| Memory-consuming user code | Large external data |  | 8 | 0 | 8 |
| Large intermediate results |  | 4 | 2 | 6 |
| Large accumulated results |  | 30 | 10 | 40 |
| Driver | Large generated results |  | 0 | 9 | 9 |
| Large collected results |  | 0 | 15 | 15 |
| Total |  |  | 56+16 | 67+1 | 123+17 |

OOM search subjects, manually review analyze causes pattern, read the comments, acquire pattern root causes interpretation, fix suggestions. Statistics. Reproduce (how to reproduce them) analyze memory usage pattern (accumulation, intermediate results)

User/system (Obtain suggestions from our reproduced jobs), Fix pattern,

Suggestions to the system designers and user.

Observations to identify the root causes. Automatically

3.1 Reproduced, profile, designed memory usage pattern, analyze user code, based on the analysis, to system designer, buffer design suggestion, user code (case: processing, sort) need a monitoring strategy, load data (3 conclusions)

3.2 automatic challenges, information, acquire information.

Summarized fix suggestions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pattern | Suggestions (Fine-grained) | Hadoop | Spark | Total |
| Large data buffers | Lower shuffle buffer size (such as Hadoop map buffer, Spark shuffle buffer) | 6 | 2 | 8 |
| Large cached data | Lower cache size |  | 2 | 2 |
| Lower storage level |  | 3 | 3 |
| Improper data partition | Add partition number, repartition | 2 | 10 | 12 |
| Change partition function |  | 1 | 1 |
| Hotspot key | Change key (e.g., add a secondary key) |  | 2 | 1 |
| Filter/skip the useless the hot key | 1 |  | 1 |
| Large single record | Split the single large record to multiple small records | 2 | 1 | 3 |
| Large external data | Decrease dataset | 1 |  | 1 |
| Large intermediate results | Read data chunk by chunk, avoid load the whole file | 1 |  | 1 |
| Decrease the input record | 1 |  | 1 |
| Large accumulated results | Add combine() to perform partial aggregation before reduce() | 2 |  | 2 |
| Aggregating/Writing the partial accumulated results onto disk periodically/continuously (before OOM, e.g., Pig spillmanager) | 2 | 1 | 3 |
| Do the operations in several passes (split the operations) (e.g., two groups in Pig, groupBy(key1).process().groupBy(key2).process()) | 3 |  | 3 |
| Move the map aggregation to reduce aggregation (e.g., from map join to reduce join) | 2 |  | 2 |
| Avoid sort by utilizing framework’s sort mechanism (e.g., secondary sort) | 3 |  | 3 |
| Reduce the input data (e.g., filter the unrelated columns in Pig Top, join) | 1 |  | 1 |
| Use application’s optimization (e.g., skew join in Pig, hive.aggr.hash.percent) | 2 |  | 2 |
| Large generated results |  |  |  |  |
| Large collected results | Remove collect()  Reduce task number (partition number)  Tree reduce, tree aggregation  Adjust the application’s parameter (e.g., lower Kmean’s K) |  | 3 | 3 |

Fix suggestions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Related pattern | Suggestions | Hadoop | Spark | Total |
| Large data buffers | Lower buffer size | 6 | 2 |  |
| Large cached data | Lower cache size |  | 2 |  |
| Lower storage level |  | 3 |  |
| Improper data partition | Add partition number, repartition | 2 | 10 |  |
| Change partition function |  | 1 |  |
| Hotspot key | Change key (add an auxiliary key) |  | 1 |  |
| Sub-divide the group, cut down the data associated to a single key |  | 2 |  |
| Aggregate partial values for each group |  | 1 |  |
| Filter/skip the useless the hot key | 1 |  |  |
| Large single record | write multiple times, split the single large record | 1 | 1 |  |
| Large external data | Decrease dataset, use reduce join | 1 |  |  |
| Large intermediate results | Add partition number, change storage |  | 1 |  |
| Read data chunk by chunk, avoid load the whole file | 1 |  |  |
| Split the single large record into multiple small records | 1 |  |  |
| Large accumulated results | Adjust the parameter, moving the array outside the iterator (for reuse), add partition number (2) | 1 | 2 |  |
| Avoid accumulation | 1 |  |  |
| Use “skew join” | 1 |  |  |
| Job split (do the operations in several passes) | 1 |  |  |
| Set hive.map.aggr.hash.percent | 1 |  |  |
| Add combiner to reduce the values for a key, in-memory combine | 2 |  |  |
| Multiple outputs (output the partial accumulated results), writing the computing results continuously | 2 |  |  |
| Use framework sort mechanism (Secondary sort) | 3 |  |  |
| Filter the unrelated columns (Pig Top) | 1 |  |  |
| Two groups (change key) | 2 |  |  |
| Output several records for the same key | 1 |  |  |
| Map partition => reduce partition |  |  |  |
| Large generates large results | Reduce partition number |  |  |  |
| Large collected results | Remove collect() (2)  Reduce task number  Tree reduce, tree aggregation  Use a small k |  |  |  |

Spark:

Caused Identified by:

|  |  |
| --- | --- |
| **Cause identified by** | **No.** |
| Users themselves | 33 |
| Experts | 29 |
| Us | 5 |
| Total | 67 |

Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sources | User-defined code | GraphX | MLlib | #Total | #Reproduced | #Unknown |
| StackOverflow | 16 | 1 | 2 | 19 | 3 | 26 |
| Spark mailing list | 42 | 1 | 5 | 48 | 14 | 80 |
| Total | 58 | 2 | 7 | 67 | 17 | 106 |

Cause patterns:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Category | Pattern | Pattern description | No. | Ratio |  |
| Large data stored in memory | Large buffered data |  | 2 |  |  |
| Large cached data |  | 7 |  |  |
| Abnormal dataflow | Improper data partition |  | 14 |  |  |
| Hotspot key |  | 7 |  |  |
| Large single record |  | 1 |  |  |
| Memory-consuming user code | Large external data |  | 0 |  |  |
| Large intermediate results |  | 2 |  |  |
| Large accumulated results |  | 10 |  |  |
| Driver | Large generated results |  | 9 |  |  |
| Large collected results |  | 15 |  |  |
| Total |  |  |  |  |  |

Fix suggestions:

|  |  |  |
| --- | --- | --- |
| Suggestions | Related pattern | No. |
| Lower buffer size | Large data buffers | 2 |
| Lower cache size | Large cached data | 2 |
| Lower storage level | 3 |
| Add partition number, repartition | Improper data partition | 10 |
| Change partition function | 1 |
| Change key (add an auxiliary key) | Hotspot key | 1 |
| Sub-divide the group, cut down the data associated to a single key | 2 |
| Aggregate partial values for each group | 1 |
| write multiple times, split the single large record | Large single record | 1 |
| Add partition number, change storage | Large intermediate results |  |
| Adjust the parameter, moving the array outside the iterator (for reuse), add partition number (2) | Large accumulated results |  |
|  | Large external data |  |
| Reduce partition number | Large generates large results |  |
| Remove collect() (2)  Reduce task number  Tree reduce, tree aggregation  Use a small k | Large collected results |  |

Hadoop:

Caused Identified by:

|  |  |
| --- | --- |
| **Cause identified by** | **No.** |
| Users themselves | 12 |
| Experts | 37 |
| Us | 7 |
| Total | 56 |

Cases:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sources | User-defined code | Pig | Hive | Mahout | Cloud9 | #Total | #Reproduced | #Unknown |
| StackOverflow | 20 | 4 | 2 | 4 |  | 30 | 15 | 32 |
| Hadoop mailing list | 5 | 5 | 1 |  | 1 | 12 | 6 | 63 |
| Developer’s blogs | 2 | 1 |  |  |  | 3 | 2 |  |
| MapReduce books | 8 | 3 |  |  |  | 11 | 2 |  |
| Total | 35 | 13 | 3 | 4 | 1 | 56 | 25 | 95 |

Cause patterns:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Category | Pattern | Pattern description | No. |  |  |
| Large data stored in memory | Large data buffers |  | 6 |  |  |
| Abnormal dataflow | Improper data partition |  | 3 |  |  |
| Hotspot key |  | 15 |  |  |
| Large single record |  | 5 |  |  |
| Memory-consuming user code | Large external data |  | 8 |  |  |
| Large intermediate results |  | 7 (5) |  |  |
| Large accumulated results |  | 30 (13) |  |  |
| Total |  |  |  |  |  |

Fix suggestions:

|  |  |  |
| --- | --- | --- |
| Suggestions | Related pattern | No. |
| Lower buffer size | Large data buffers | 6 |
| Add reduce number | Improper data partition | 2+1 |
| Filter/skip the useless hot key | Hotspot key | 1 |
| Read data chunk by chunk, avoid load the whole file | Large single record | 1 |
| Break up/split the large record into smaller records | Large single record  Large intermediate results | 2 |
| Change code logic (e.g., use reduce join ) | Large external data |  |
| Code (avoid accumulation),  Use skew join,  <k, v> => <(k1, k2), v>  Job split, operation to several passes,  Set hive.map.aggr.hash.percentmemeory  Add combiner (2)  Multiple outputs  Data filter (remove unrelated columns)  Use secondary sort (3)  Group by, filter, groupBy (Two groups)  Avoid in-memory aggregation  Output several records for the same key  Spill the results onto disk continuously | Large accumulated results |  |