Hadoop MapReduce

How to Survive Out-of-Memory Errors

The international Summer Undergraduate Research Fellowship

Abstract

As modern computing enters the era of Big Data, Hadoop MapReduce is a widely-adopted programming model for large scale data processing due to its simple interface. However, often data-intensive applications fail to achieve satisfactory performance and/or suffer program crashes due to out-of-memory errors. We have conducted an extensive study by gathering and analyzing real-world performance cases submitted by StackOverflow users. The study shows that performance problems are common in Big Data systems. We also find through the study that while there are many reasons for poor performance of Big Data applications, practical solutions always center around making "best practice" recommendations for tuning of framework parameters. Unfortunately, framework such as Hadoop has numerous parameters which interact in complicated ways. Our research aims to identify most impactful parameters and understand their contribution on applications' performance. We develop a set of in-house data-intensive programs as well as several real-world business applications. These applications are then heavily simulated under various configurations in a cluster. We have found two important parameters and performed rigorous stress-test on them. Our experimental results provide strong empirical evidence for a near-optimal configuration under which many Big Data applications have better performance and/or survive crashes. We also have distilled valuable insights that can aid non-expert developers in their performance tuning process.

1. Introduction

"There was 5 exabytes of information created between the dawn of civilization through 2003, But that much information is now created every 2 days, and the pace is increasing...". Eric Schmidt, CEO of Google spoke at the '10 Techonomy conference in Lake Tahoe. Our generation now enters the era of Big Data, and Big Data analyzing technologies are become more important.

Hadoop MapReduce is one of the mainly used Programming model for Big Data. This model aims that programmer can implement parallel-processing program very easily, even they didn't experience about distributed systems. They don't need to understand the details of parallelization, fault tolerance, data distribution and load balancing.

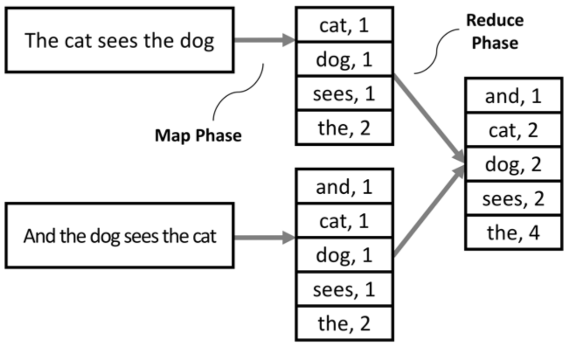

Figure 1: Example of MapReduce Workflow

Figure 1 shows the MapReduce Workflow. The name of 'MapReduce' is combination of the word 'Map' and the word 'Reduce'. The Map Task Worker called 'Mapper' takes the spited input files called 'Map Tasks'. When the single map task ends, Mapper emits the intermediate key/value pairs. And then, Reducer worker called 'Reducer' take these intermediate key/value pairs, and make an output file. This file is quite simple to read for humans.

But Hadoop MapReduce Program sometimes suffer the Memory Pressure, which make poor performances and even make Out-of-Memory which crashes the program. There are many reasons for these problems, but even experts sometimes can't figure them out. Our research aims to identify the reason of these problems by gathering and analyzing real-world performance cases submitted by *StackOverflow.com* users. We investigate two reasons why OOM occurs:

Inappropriate Configuration We ran Apache Hadoop version 1.0.3. After program ran, log files are created with output file. One of the log is about configuration parameters and it has 231 lines. We estimate that if these numerous parameters are inappropriate, System performance will be decreased.

Large Intermediate Results We found that some program's Mapper emits too large intermediate key/value pairs. These too large intermediate data structure exceeds entire JAVA Heap Space. Some cases shows that Some keys called 'Hot Keys' which has too high occurrence ratio exceeds the Heap Space.

To investigate these poor performance problems, we heavily simulated under various configurations with three kinds of operations: Standalone, Pseudo-distributed and Fully distributed. We used our own laptop for Standalone and Pseudo-distributed operation. And we constructed Raspberry Pi Three-node cluster for Fully-distributed operations.

Summary of results After rigorous stress-test, we have found some parameters which associate with Performances. And we found two important parameters: Split Size and the number of Map/Reduce worker capacity. These parameters highly influence the program's performance.
And our study focused on some large intermediate result problems. We choose one Java Heap error case and heavily simulated under various configurations. By this study, we could measure the size of intermediate data structure, and found some solutions for this case.

2. MapReduce Programming Model

MapReduce programming model has several advantages. First, MapReduce Programming model is very easy to use.
Distribute processing has several concepts like parallelization, fault tolerance, data distribution, and load balancing. Detail of these concepts are hard to understand within a short time. But MapReduce programming model hides the detail of these concepts. So, programmer who even had no experiences about distribute processing can easily use this programming model.
The second advantage is that a large variety of Real-world problems are easily expressible in MapReduce Programing model. There are a lot of examples like word count, standard deviation and join operation for large scale datasets. And these example is easily expressed by MapReduce Programming model.
The last advantage is that clusters can be easily scales.

So, we choose the MapReduce programming model for research Distribute processing.

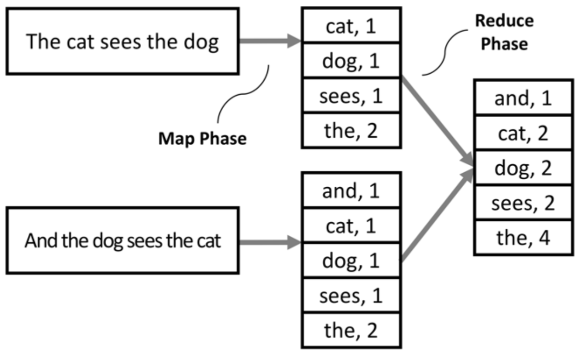

Figure 1. Example of Execution Overflow

The name ‘MapReduce’ is combination of the word ‘Map’ and the word ‘Reduce.’ The programing model also has two phase: ‘Map’ phase and ‘Reduce’ phase.

A

3. Hadoop MapReduce Installation

a

Single-node Operation

a

Pseudo-distributed Operation

a

Fully-distributed Operation

a

number of mapper

4. Cause of Poor Performance

we found several important parameters

Split Size

a

io.sort.mb

a

5. OOM Problem Investigation

Stackoverflow.com OOM case simulation

Java Heap Size

A

Number of Map task worker

A

Solution

a

5. Conclusion

6. References

Figure 1: Example of Map Workflow

Figure 2: Example of Reduce Workflow