An Analysis of Side Effects of Hadoop's Optimization in Data Center

1. Introduction

Motivation

* 1. Objective
  2. Contribution
  3. Results
  4. Thesis Overview

1. Background
   1. MapReduce
   2. Architecture of Hadoop
      1. HDFS
      2. Hadoop Components
      3. FIFO Scheduler
   3. Hadoop Optimizations
      1. Capacity Scheduler
      2. Fair Share Scheduler
      3. LATE Scheduler
      4. Dynamic Scheduler
      5. Delay Scheduler
2. Methodology
3. Experimental Environment
   1. TestBed
   2. Hadoop
      1. Hadoop Scheduler
      2. Hadoop Component
   3. Ubuntu VM
   4. Workload
      1. TeraSort
      2. Facebook Synthesize Traces
   5. Open Stack
   6. Resources
      1. Physical Machines
      2. Component
   7. Tools
      1. Bandwidth Monitor
      2. MPstat
      3. Iostat
   8. Summary
4. Evaluation
   1. Hadoop Base line
      1. Objective
      2. Experiment Description
      3. Discussion
   2. Hadoop Colocated Data Nodes
      1. Objective
      2. Experiment Description
      3. Discussion
   3. Hadoop Colocated Clusters
5. Conclusion
   1. Discussion
   2. Future Work

Bibliography

**Background**

Today's most popular applications are interned based applications such as, social media, e-commerce, etc. For user interacts with these applications , various data such as clickstream data,crawled web documents, web requests, logs, etc, are generated. As the applications serves millions of users around the globe, so the amount generated data is also huge or so called Big Data. This Big Data is is potential gold mine for the companies to understand access pattern and ad revenue of the company. clickstream data for user actions are the main sources for the developers and operators to diagnose problems in production.

The Authors in Google implemented many special-purpose computation paradigm in the past years. The purpose of these spcial-purpose computation was to process large amount of raw data such as crawled web documents, web requests, logs, etc. The process of large data helps Google to compute various graph of derived data, such as inverted indices, various graph representation of web documents, summaries of number of pages crawled per host, the set of most frequent queries in a data , etc.

1. **MapReduce**

MapReduce is a programming model and associated implementation for process large data sets in parallel. The mapreduce program reads input key/value pairs and generates output key/value pairs. A mapreduce program consist of Map and Reduce phases. The map and reduce phases can be defined as map and reduce functions written by programmer. The map function read input data as key/value pairs and generates intermediate values.Output of map function si processed by mapreduce platform to The reduce function reads intermediate data generated from map function(s) and merges all values associated with same intermediate key.

Mapreduce is designed to run jobs that lasts minutes or hours on dedicated hardware in single data center, with very high bandwidth interconnects.

\*\*\* Map Reduce Example, page 21, Hadoop Definitive Guide, 3rd Edition.

1. **Hadoop**

Hadoop was created by Doug Cutting, the creator of Apache Lucene, the widely used text search library[3].The name hadoop is not an acronyme; it is made-up name by kid of Doug Cutting calling his yellow elephant toy “hadoop”.

**Job:** MapReduce job is unit of work that need to be processed by nodes. It consist of input data, job configuration information and MapReduce program. To run the jobs, Hadoop divides it into smaller pieces called tasks. There are two types of tasks: map tasks and reduce tasks.

Job execution are controlled by two components of Hadoop called JobTracker and TaskTracker. Jobtracker is responsible to run all jobs on system. It coordinates job execution by scheduling tasks on tasktracker. The jobtracker maintains record about status of each job. The tasktracker executes tasks on nodes and sends progress report to jobtracker. In case if task execution failed, the jobtracker can rescheduler the task on same or different tasktracker.

**Job Creation:** Usually the data to be process by Hadoop is very large data set. Hadoop divides this large input data to small fixed-size “input split” or “split”. Each split is called MapReduce job. The splits are fed as input to user defined map function. Map functions read each record of input split and process it. Having many small jobs mean that the execution time for each job is smaller comparing to large input. So if we run the small jobs in parallel on Hadoop cluster, the total time of processing all small jobs will be smaller than total time to process large input data set. [1]

If the job size is very small , then the job creation and map creation time will dominante the over all execution of job. Therefor usually the size of MapReduce jobs are the same as the HDFS block size , which is 64 MB. Such job size is good for Rack Locality Feature of Hadoop.

**Data Locality:**