DYNAMIC RESOURCE ALLOCATION FOR MAPREDUCE WITH PARTITIONING SKEW

ABSTRACT

MapReduce has become a prevalent programming model for building data processing applications in the cloud. While being widely used, existing MapReduce schedulers still suffer from an issue known as partitioning skew, where the output of map tasks is unevenly distributed among reduce tasks. Existing solutions follow a similar principle that repartitions workload among reduce tasks. However, those approaches often incur high performance overhead due to the partition size prediction and repartitioning. In this paper, we present DREAMS, a framework that provides run-time partitioning skew mitigation. Instead of repartitioning workload among reduce tasks, we cope with the partitioning skew problem by controlling the amount of resources allocated to each reduce task. Our approach completely eliminates the repartitioning overhead, yet is simple to implement. Experiments using both real and synthetic workloads running on a 21-node Hadoop cluster demonstrate that DREAMS can effectively mitigate the negative impact of partitioning skew, thereby improving the job completion time by up to a factor of 2.29 over the native Hadoop YARN. Compared to the state-of-the-art solution, DREAMS can improve the job completion time by a factor of 1.65.

EXISTING SYSTEM:

Several recent approaches are proposed to handle the partitioning skew problem .They follow a similar principle that predicts the workload for individual reduce tasks based on certain statistics of key-value pairs (*e.g.* key frequencies), and then repartitions the workload to achieve a better balance among the reduce tasks. However, in order to collect the statistics of keyvalue pairs, most of those solutions either have to prevent the reduce phase from overlapping with the map phase, or add a sampling phase before executing the actual job. Skewtune can reduce this waiting time by redistributing the unprocessed workload of a slow reduce task at runtime. However, Skewtune incurs an additional run-time overhead of approximately 30 seconds (as reported in). This overhead can be quite expensive for small jobs with average life span of around 100 seconds, which are very common in today's production clusters .

DISADVANTAGES:

- The completion time of a MapReduce job is determined by the completion time of the slowest reduce task.
- Data skewness causes certain tasks with heavy workload run slower than others.

PROPOSED SYSTEM:

- We first develop a partition size prediction model that can forecast the partition sizes of reduce tasks at run-time. Specifically, we can accurately predict the size of each partition when only 5% of map tasks have completed.
- We establish a task performance model that correlates the completion time of individual reduce tasks with their partition sizes and resource allocation.
- We propose a scheduling algorithm that dynamically adjusts resource allocation to each reduce task using our task performance model and the estimation of the partition size. This can reduce the running time difference among reduce tasks that have different sizes of partitions to process, thereby accelerating the job completion.

ADVANTAGE OF PROPOSED SYSTEM:

- It can improve the job completion time by a factor of 1.65
- It can forecast the partition sizes of reduce tasks at run-time.
- This can reduce the running time difference among reduce tasks that have different sizes of partitions to process, thereby accelerating the job completion.

SYSTEM SPECIFICATION

Hardware Requirements:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Sony.
- Ram : 512 Mb.

Software Requirements:

- Operating system : Windows 7.
- Coding Language : ASP.Net with C#
- Data Base : SQL Server 2005.